Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη και Σχεδίαση ενός Έξυπνου-Crowdsensing Ταχύμετρου για Android** **Design and Develop a Smart-Crowdsensing Speedometer for Android**
Ονοματεπώνυμο Φοιτητή	**Ευφροσύνη Σιγάλα**
Πατρώνυμο	**Μιχαήλ**
Αριθμός Μητρώου	**ΜΠΠΛ / 15064**
Επιβλέπων	**Επ. Καθηγητής Αλέπης Ευθύμιος**

Ημερομηνία Παράδοσης **Νοέμβριος 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα επίκουρο καθηγητή κ.Ευθύμιο Αλέπη, για την ευκαιρία που μου έδωσε να επιλέξω αυτή τη μεταπτυχιακή διατριβή και για την πολύτιμη συμβολή του στην ολοκλήρωσή της, καθώς και για τις γνώσεις που μου μετέδωσε κατά τη φοίτησή μου στο μεταπτυχιακό πρόγραμμα της Πληροφορικής.

Τέλος, θα ήθελα να ευχαριστήσω ιδιαιτέρως την οικογένειά μου και τους στενούς μου ανθρώπους για την ακατάπαυστη ενθάρρυνση και υποστήριξή τους, καθ' όλη τη διάρκεια των σπουδών μου.

Περίληψη

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής είναι η σχεδίαση και ανάπτυξη μιας εφαρμογής, η οποία συλλέγει δεδομένα από τους αισθητήρες των κινητών συσκευών (έξυπνων κινητών τηλεφώνων και ταμπλετών) με λειτουργικό σύστημα Android. Οι εφαρμογές που συλλέγουν δεδομένα κατ' αυτόν τον τρόπο, ονομάζονται crowdsensing εφαρμογές. Η συλλογή των δεδομένων είναι ευκαιριακή, όπου οι χρήστες συμβάλλουν έμμεσα στη συλλογή των πληροφοριών, εκκινώντας απλώς την εφαρμογή.

Τα δεδομένα αφορούν την τοποθεσία του χρήστη, την ταχύτητα, καθώς και τους κραδασμούς που προκαλούνται κατά τη μετακίνησή του, με σκοπό την περαιτέρω ανάλυσή τους (Big Data Analytics) στο μέλλον, εξάγοντας χρήσιμες πληροφορίες σχετικά με τις συνθήκες κυκλοφοριακής συμφόρησης και την ποιότητα του οδοστρώματος ανά περιοχή, σε πραγματικό χρόνο.

Η εφαρμογή προσομοιώνει τη λειτουργία ενός ταχύμετρου, χρησιμοποιώντας είτε το ενσωματωμένο σύστημα GPS είτε τα δίκτυα της συσκευής. Υπολογίζει την επιτάχυνση, τη μέση και τη μέγιστη ταχύτητα, τη μέση και τη μέγιστη επιτάχυνση, το συνολικό χρόνο και τη διανυθείσα απόσταση. Επιπρόσθετα, εμφανίζει την τρέχουσα τοποθεσία του χρήστη πάνω στο χάρτη, καθώς και το ιστορικό των διαδρομών του.

Abstract

This thesis focuses on the design and development of an application that collects data from the sensors of mobile devices (smartphones and tablets) with Android operating system. Apps which collect data in this specific way are called crowdsensing applications. Data collection is opportunistic where users contribute indirectly to the collection of information, by simply starting the app.

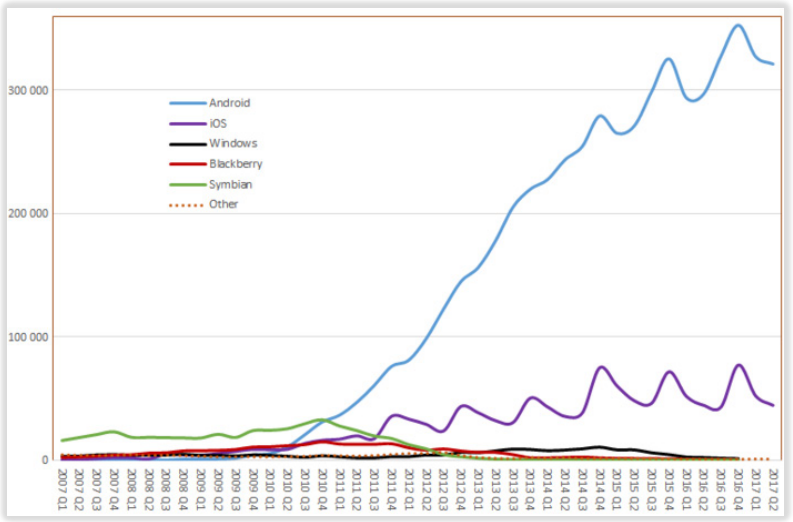
The data relates to the location of the user, the speed and the shakes caused by his movement, for purpose of further analysis (Big Data Analytics) in the future, by extracting useful information on congestion conditions and road surface quality, in real time.

The application simulates the function of a speedometer using either the built-in GPS system or the device's networks. Calculates acceleration, average and maximum speed, average and maximum acceleration, total time and the distance traveled. In addition, displays the user's current location on the map, as well as the history of his routes.

Περιεχόμενα

Εισαγωγή

Οι έξυπνες συσκευές (smartphones) έχουν γίνει αναπόσπαστο κομμάτι της ζωής μας. Συγκεκριμένα, τα smartphones με λειτουργικό σύστημα Android έλαβαν το 86,1% μερίδιο αγοράς το πρώτο τρίμηνο του 2017 (“Worldwide Sales of Smartphones”, 2017). Σημαντικό ρόλο σε αυτό συνετέλεσε πως οι συσκευές Android διατίθενται σε μεγάλο εύρος τιμών, καλύπτοντας καταναλωτές με διαφορετική αγοραστική δύναμη. Παρακάτω φαίνεται ένα διάγραμμα πωλήσεων έξυπνων συσκευών παγκοσμίως (“Mobile Operating System”, 2017):



Εικόνα 1 Παγκόσμιες πωλήσεις έξυπνων συσκευών (σε χιλιάδες).

Σήμερα, όλες οι συσκευές Android μπορούν να συνδεθούν στο **διαδίκτυο**. Επιπρόσθετα, οι περισσότερες από αυτές, διαθέτουν ισχυρούς επεξεργαστές και πλήθος αισθητήρων (sensors), όπως **ενσωματωμένο σύστημα GPS (Global Positioning System)** και αισθητήρα **επιταχυνσιόμετρου (accelerometer)**, δίνοντας στους προγραμματιστές τη δυνατότητα να αναπτύξουν πλήθος εφαρμογών, αξιοποιώντας τα δεδομένα που λαμβάνουν από αυτούς.

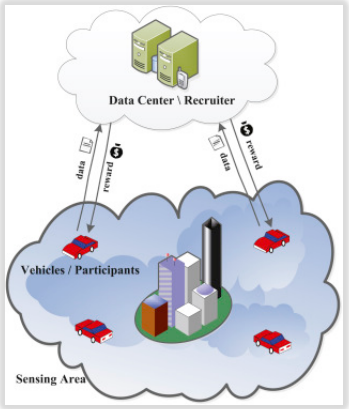
Εικόνα 2 Αισθητήρες σε έξυπνη συσκευή Android.

Μέσω των δύο προαναφερθέντων αισθητήρων μπορεί να γίνει συλλογή τεράστιου όγκου δεδομένων (Big Data), τα οποία, σε περαιτέρω μελέτη, μπορούν να αναλυθούν μέσω κατάλληλων τεχνικών (Big Data Analytics), οδηγώντας στην εξαγωγή χρήσιμων πληροφοριών (Ganti & Ye & Lei, 2011). Μέσω των πληροφοριών αυτών, μπορούν **να παραχθούν χάρτες με τις συνθήκες κυκλοφοριακής συμφόρησης, καθώς και την ποιότητα του οδοστρώματος ανά περιοχή, σε πραγματικό χρόνο (real-time)**, παράγοντας δηλαδή νέα γνώση. Με την αξιοποίηση της γνώσης αυτής, οι χρήστες θα μπορούν να αποφεύγουν τους δρόμους με κυκλοφοριακή

συμφόρηση και οι δήμοι να ενημερώνονται αυτόματα για τις κακοτεχνίες του οδοστρώματος, προβαίνοντας σε άμεση αποκατάστασή τους. Οι τεχνικές αυτές έχουν ως απώτερο στόχο τη βελτίωση της μετακίνησης των πολιτών, που συνεπάγεται με τη βελτίωση της ποιότητας της ζωής τους.


Εικόνα 3 Κυκλοφορική συμφόρηση και κακοτεχνίες οδοστρώματος.

Σύμφωνα με δημοσίευση των Jian, Xiaolin, Jianwei, Yu και Xin στο Διεθνές Συνέδριο IEEE «Multimedia Big Data (BigMM)», που διεξήχθη τον Απρίλιο του 2015 στην Κίνα, η συλλογή πληροφοριών, μέσω των ενσωματωμένων αισθητήρων που διαθέτουν οι έξυπνες συσκευές, είναι μία νέα τάση ανάπτυξης στο Δίκτυο των Πραγμάτων (IoT - Internet of Things), γνωστή ως **Mobile Crowd Sensing (MCS)**. Η μέθοδος MCS παρουσιάζει πολυάριθμες και μοναδικές ερευνητικές προκλήσεις, καθότι η ανθρώπινη συμμετοχή επηρεάζει τα δεδομένα. Τα δεδομένα εξαρτώνται από την περιοχή στην οποία μένει και κινείται ο χρήστης, από τα κίνητρά του και από την παροδική σύνδεση στο διαδίκτυο. Η συλλογή δεδομένων μπορεί είναι είτε **συμμετοχική (participatory)**, δηλαδή οι χρήστες συμμετέχουν ενεργά στην παραγωγή δεδομένων, κάνοντας για παράδειγμα οι ίδιοι αναφορά για κάποια κακοτεχνία του δρόμου, είτε **ευκαιριακή (opportunistic)**, όπου οι χρήστες συμβάλλουν έμμεσα στη συλλογή πληροφοριών. Επιπλέον, εγείρονται μοναδικά ζητήματα σχετικά με την ιδιωτικότητα και την ασφάλεια των δεδομένων, ως ευαίσθητες πληροφορίες, όπως η τοποθεσία του χρήστη (Mobile Crowd Sensing, 2017). Τέλος, αναγκαίος είναι ο έλεγχος της ποιότητας και της αξιοπιστίας των δεδομένων, από πλαστά δεδομένα (counterfeit data) που παραχωρούν κακόβουλοι χρήστες (Tanas & Herrera, 2013).



Εικόνα 4 Αρχιτεκτονική συστημάτων Mobile Crowd Sensing.

Στο χώρο των κινητών συσκευών, οι εφαρμογές Android υποστηρίζονται από την **Google** και το κόστος ανάπτυξής τους είναι ιδιαίτερα χαμηλό, σε σχέση με τις εφαρμογές για iOs (Apple) και Windows Mobile (Microsoft), καθώς τα περισσότερα εργαλεία διατίθενται δωρεάν ή με χαμηλό κόστος. Οι εφαρμογές Android χωρίζονται σε τρεις κύριες κατηγορίες:

- **Web apps**: Μπορούν να αναπτυχθούν εύκολα και γρήγορα, με πολύ χαμηλό κόστος. Υστερούν σε επιδόσεις, δεν υποστηρίζουν άμεσα τις πιο σύγχρονες τεχνολογίες και το User Experience είναι ιδιαίτερα χαμηλό.

- **Hybrid apps**: Προσφέρουν περισσότερα χαρακτηριστικά από τα Web apps, με καλύτερη εμπειρία χρήσης, όχι όμως τη βέλτιστη, με μέσο κόστος ανάπτυξης. Τα Hybrid apps, όπως και τα Web apps, δεν προσφέρουν άμεσα τις τελευταίες τεχνολογίες.
- **Native apps:** Προσφέρουν το καλύτερο User Experience, βέλτιστη χρηστικότητα και είναι ταχύτατα. Επίσης, υποστηρίζουν άμεσα τις νέες τεχνολογίες του λειτουργικού Android, παρέχοντας τη δυνατότητα αναβάθμισης σε αυτές. Το κόστος ανάπτυξης είναι λίγο υψηλότερο σε σχέση με τις άλλες δύο κατηγορίες, καθώς απαιτούνται περισσότερες εργατοώρες για την ανάπτυξη του λογισμικού, αλλά προσφέρουν τις καλύτερες επιδόσεις.

Για τους λόγους που προαναφέρθηκαν, ως αντικείμενο της παρούσας μεταπτυχιακής διατριβής επελέγη η σχεδίαση και η ανάπτυξη μιας **native, crowdsensing εφαρμογής**, στοχεύοντας σε έξυπνα κινητά τηλέφωνα και ταμπλέτες, **με λειτουργικό σύστημα Android**, όπου ο χρήστης, εν γνώσει του, συμβάλλει έμμεσα στη συλλογή πληροφοριών από τους αισθητήρες της συσκευής του (human crowdsensing - opportunistic), εκκινώντας απλώς την εφαρμογή. Η εφαρμογή έχει τις ακόλουθες δυνατότητες:

- ✓ Συλλέγει δεδομένα που αφορούν την τοποθεσία του χρήστη, την ταχύτητα με την οποία κινείται, καθώς και τους κραδασμούς που προκαλούνται κατά τη μετακίνησή του. Όταν η σύνδεση με το διαδίκτυο είναι ενεργοποιημένη, τα δεδομένα αποστέλλονται με ασφάλεια σε βάση δεδομένων στο διαδίκτυο και δε διαμοιράζονται σε τρίτους, κρατώντας ασφαλή τα προσωπικά δεδομένα του χρήστη.
- ✓ Χρησιμοποιώντας είτε το ενσωματωμένο σύστημα GPS είτε τα δίκτυα της συσκευής, εμφανίζει την ταχύτητα του χρήστη, την επιτάχυνση, τη μέση και τη μέγιστη ταχύτητα, τη μέση και τη μέγιστη επιτάχυνση, το συνολικό χρόνο και τη διανυθείσα απόσταση. Επιπρόσθετα, εμφανίζει την τρέχουσα τοποθεσία του χρήστη πάνω στο χάρτη, συμπεριλαμβάνοντας την οδό και τις συντεταγμένες, καθώς και το ιστορικό των διαδρομών του.
- ✓ Είναι σχεδιασμένη με τέτοιο τρόπο, ούτως ώστε να καλύπτει το μεγαλύτερο εύρος έξυπνων συσκευών και ταμπλετών που κυκλοφορούν στην αγορά, χρησιμοποιώντας διαφορετικούς πόρους (resources) για διαφορετικές διαστάσεις οθονών και διαφορετικές εκδόσεις λειτουργικών συστημάτων Android.
- ✓ Διαθέτει πολύγλωσση υποστήριξη, με διαθέσιμες γλώσσες τα Ελληνικά και τα Αγγλικά.
- ✓ Διατηρεί την οθόνη ανοιχτή, χωρίς να αφήνει τη συσκευή να μεταβεί σε κατάσταση ύπνου, όσο βρίσκεται στο προσκήνιο.
- ✓ Στον κώδικα αποθηκεύονται κατάλληλα όλες οι μεταβλητές, ώστε να μη χάνονται κατά την αλλαγή προσανατολισμού.
- ✓ Γίνονται όλοι οι απαραίτητοι έλεγχοι, ώστε να μην προκαλούνται σφάλματα κατά τη λειτουργία της.

Η εφαρμογή έχει την ονομασία **«Ταχύμετρο» (Speedometer)** και διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

https://play.google.com/store/apps/details?id=net.bplaced.esigala1.speedometer

Εναλλακτικά, μπορείτε να σαρώσετε τον κωδικό QR που βρίσκεται πιο κάτω, μέσω της κινητής σας συσκευής, ώστε να μεταβείτε απευθείας στην σελίδα του Ταχύμετρου στο Play Store:

ΚΕΦΑΛΑΙΟ 1: Ανασκόπηση Πεδίου

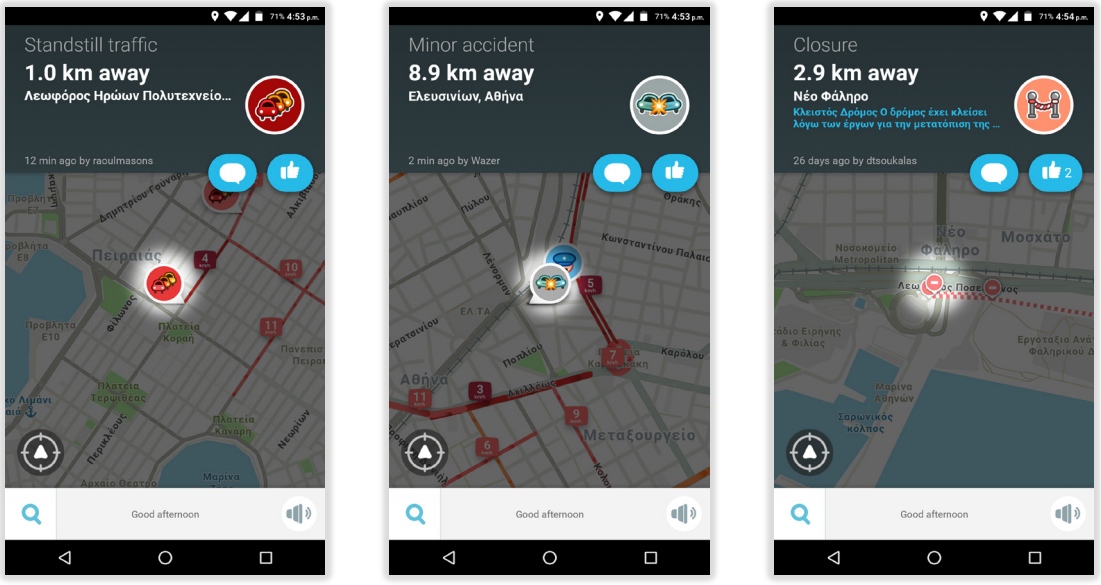
Στο Play Store μπορούμε να βρούμε πλήθος εφαρμογών, από όλο τον κόσμο, που κάνουν χρήση των αισθητήρων τοποθεσίας και επιταχυνσιόμετρου των κινητών συσκευών, είτε συλλέγοντας τα δεδομένα αυτά (crowdsensing εφαρμογές) είτε απλώς προβάλλοντας τις μετρήσεις στην οθόνη της συσκευής.

1.1 Εφαρμογές Εξωτερικού

1.1.1 Waze - GPS, Maps, Traffic Alerts & Sat Nav (Waze)

Η εφαρμογή «Waze - GPS, Maps, Traffic Alerts & Sat Nav» προσφέρεται από την ομάδα Waze, η οποία αποτελείται από προγραμματιστές, καθώς και επαγγελματίες διαφόρων άλλων ειδικοτήτων.

- Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:
  - Προβολή της τοποθεσίας του χρήστη στο χάρτη.
  - Οδηγίες πλοήγησης.
  - Ανίχνευση κυκλοφοριακής συμφόρησης.
  - Αποφυγή δρόμων με κυκλοφοριακή συμφόρηση.
  - Ειδοποιήσεις σχετικά με την κυκλοφορία, την αστυνομία και διάφορα έκτακτα περιστατικά που συμβαίνουν στους δρόμους.



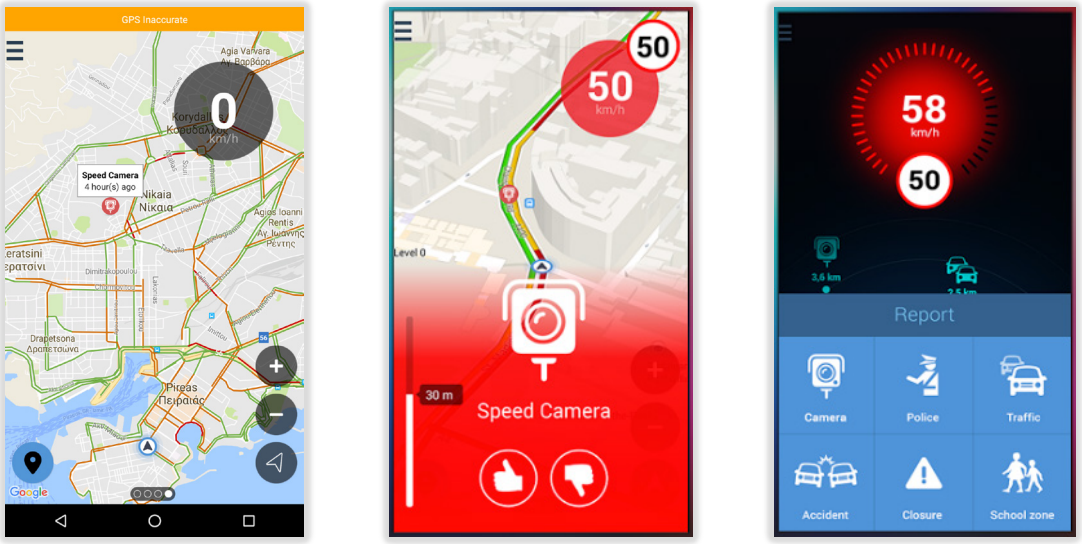
Εικόνα 5 Στιγμιότυπα από την εφαρμογή «Waze - GPS, Maps, Traffic Alerts & Sat Nav» (Waze).

Η εφαρμογή «Waze - GPS, Maps, Traffic Alerts & Sat Nav» διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:
https://play.google.com/store/apps/details?id=com.waze

1.1.2 Speed Cameras & Traffic Sygic (Sygic)

Η εφαρμογή «Speed Cameras & Traffic Sygic» προσφέρεται από την εταιρία προγραμματισμού Sygic, η οποία αναπτύσσει εφαρμογές πλοήγησης για κινητές συσκευές.

- Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:
  - Προβολή της τοποθεσίας του χρήστη στο χάρτη.
  - Ανίχνευση ορίων ταχύτητας.
  - Λήψη ειδοποιήσεων σε πραγματικό χρόνο σχετικά με ραντάρ και κάμερες ανίχνευσης ταχύτητας. Ο χρήστης έχει τη δυνατότητα να στείλει ενημερώσεις στη διαδικτυακή βάση δεδομένων, σχετικά με νέες τοποθεσίες καμερών ανίχνευσης ταχύτητας.
  - Ηχητικές ειδοποιήσεις όταν ο χρήστης ξεπερνά το όριο ταχύτητας.
  - Καταγραφή ταχύτητας και άλλων στατιστικών στοιχείων σχετικά με την οδήγηση.



Εικόνα 6 Στιγμιότυπα από την εφαρμογή «Speed Cameras & Traffic Sygic» (Sygic).

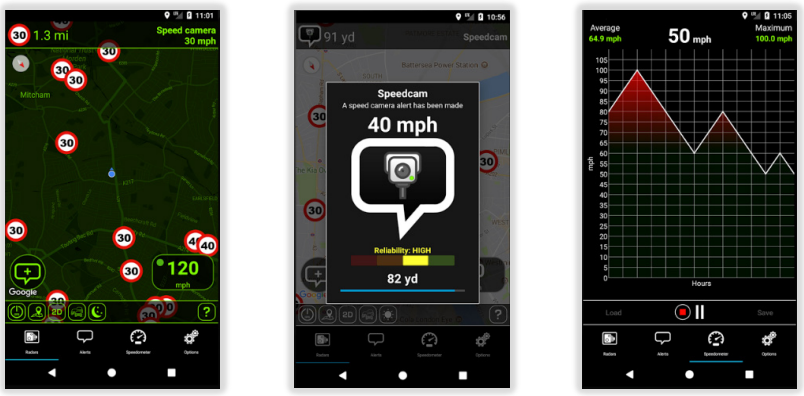
Η εφαρμογή «Speed Cameras & Traffic Sygic» διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:
https://play.google.com/store/apps/details?id=com.sygic.speedcamapp

1.1.3 Radarbot Free: Speed Camera Detector & Speedometer (Vialsoft)

Η εφαρμογή «Radarbot Free: Speed Camera Detector & Speedometer» προσφέρεται από την εταιρία προγραμματισμού Vialsoft.

- Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:
  - Προβολή της τοποθεσίας του χρήστη στο χάρτη.
  - Καταγραφή της ταχύτητας του χρήστη και ανίχνευση ορίων ταχύτητας.
  - Ανίχνευση καμερών ταχύτητας.
  - Ηχητικές προειδοποιήσεις.

- Νυχτερινή λειτουργία.



Εικόνα 7 Στιγμιότυπα από την εφαρμογή «Radarbot Free: Speed Camera Detector & Speedometer» (Vialsoft).

Η εφαρμογή «Radarbot Free: Speed Camera Detector & Speedometer» (Vialsoft) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

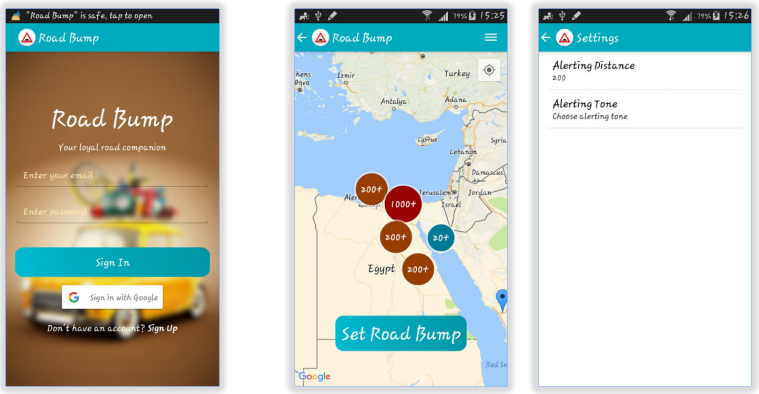
https://play.google.com/store/apps/details?id=com.vialsoft.radarbot_free

1.1.4 Road Bump (Mostafa M.N.)

Η εφαρμογή «Road Bump» προσφέρεται από τον προγραμματιστή Mostafa M.N. και συλλέγει πληροφορίες σχετικά με τις κακοτεχνίες που υπάρχουν στους δρόμους. Ο χρήστης αρκεί να ενεργοποιήσει τα δεδομένα και την τοποθεσία της συσκευής του και η εφαρμογή αυτόματα αποστέλλει τα δεδομένα που συλλέγει από τους κατάλληλους αισθητήρες.

❖ Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:

- Προβολή της τοποθεσίας του χρήστη στο χάρτη.
- Χρήση τοπικής και διαδικτυακής βάσης δεδομένων.
- Ηχητική προειδοποίηση όταν ο χρήστης προσεγγίζει ένα σημείο που έχει οριστεί ως κακοτεχνία δρόμου.
- Σύστημα ελέγχου ταυτότητας για την εξασφάλιση αξιόπιστων δεδομένων.



Εικόνα 8 Στιγμιότυπα από την εφαρμογή «Road Bump» (Mostafa M.N.).

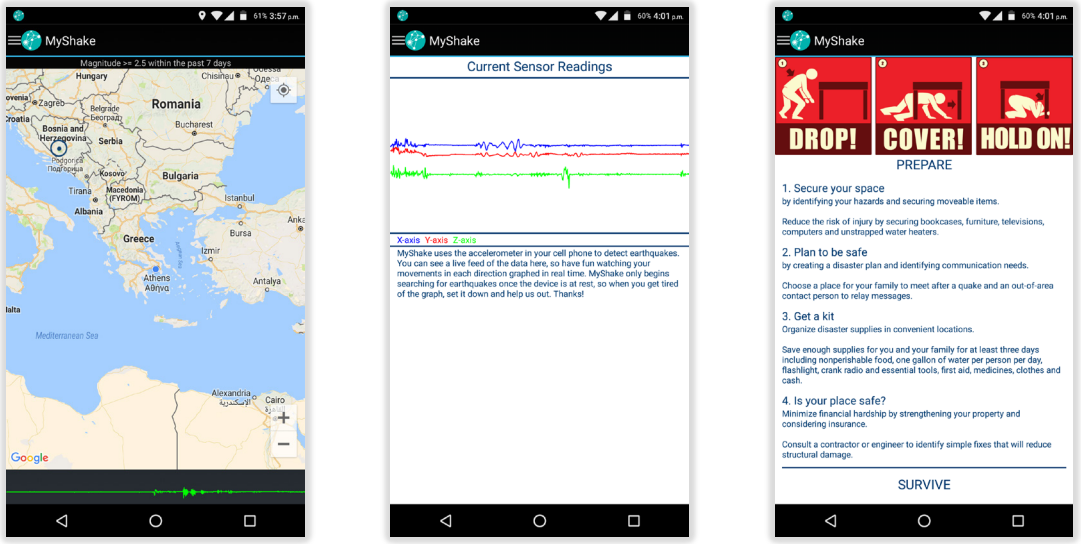
Η εφαρμογή «Road Bump» (Mostafa M.N.) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

https://play.google.com/store/apps/details?id=com.mostafa.android.roadbump

1.1.5 MyShake (UC Berkeley Seismological Laboratory)

Η εφαρμογή «MyShake» προσφέρεται από το σεισμολογικό εργαστήριο Berkeley που εδρεύει στη Νότια Καλιφόρνια. Καταγράφει τις σεισμικές δονήσεις και μελλοντικά στοχεύει στην πρόβλεψη των σεισμών, αποστέλλοντας στους χρήστες προειδοποιήσεις πριν αυτοί συμβούν. *Η εφαρμογή αυτή θα μπορούσε κάλλιστα να χρησιμοποιηθεί για τον εντοπισμό των κακοτεχνιών στους δρόμους, αφού συλλέγουν αντίστοιχα δεδομένα.*

- Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:
  - Προβολή της τοποθεσίας του χρήστη στο χάρτη.
  - Γραφική απεικόνιση των δονήσεων της συσκευής.
  - Οδηγίες που πρέπει να ακολουθήσουμε σε περίπτωση σεισμού.



Εικόνα 9 Στιγμιότυπα από την εφαρμογή «MyShake» (UC Berkeley Seismological Laboratory).

Η εφαρμογή «MyShake» (UC Berkeley Seismological Laboratory) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

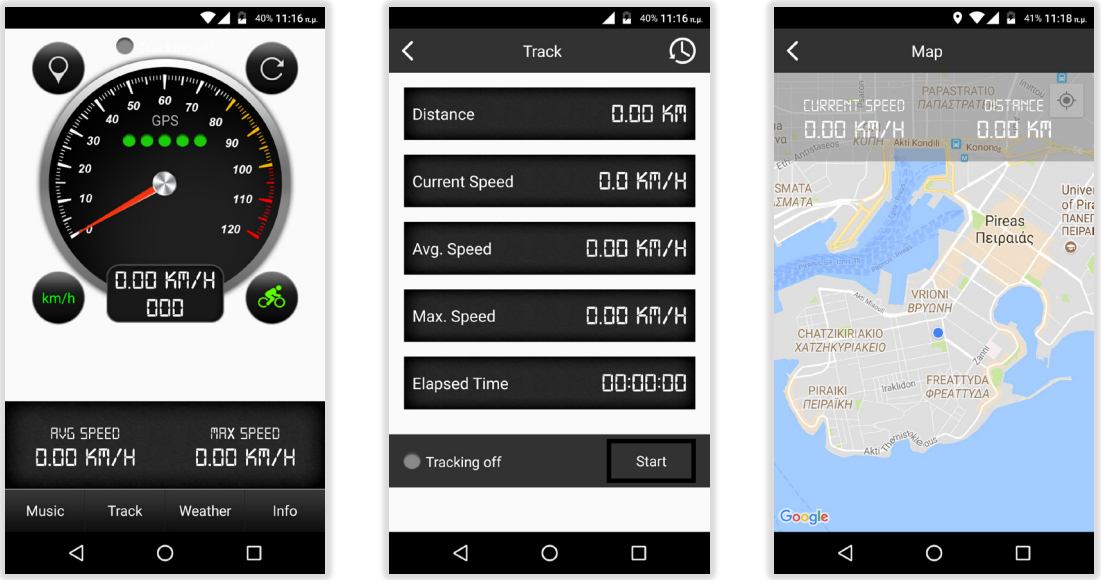
https://play.google.com/store/apps/details?id=edu.berkeley.bsl.myshake

1.1.6 GPS Speedometer (Guru Info Media)

Η εφαρμογή «GPS Speedometer» προσφέρεται από την εταιρία Guru Info Media. Η εταιρία αυτή εδρεύει στην Ινδία και ειδικεύεται στην ανάπτυξη εφαρμογών για κινητές συσκευές με λειτουργικό σύστημα Android και iOS, καθώς και στη σχεδίαση ιστοσελίδων.

- Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:
  - Ένδειξη της τρέχουσας ταχύτητας σε ψηφιακό και αναλογικό ταχύμετρο.

- Παρακολούθηση διανυθείσας απόστασης, χρόνου, μέγιστης και μέσης ταχύτητας.
- Εναλλαγή μεταξύ μονάδων μέτρησης (mph ή km/h).
- Επιλογή τύπου ταξιδιού (αυτοκίνητο, ποδήλατο).
- Εμφάνιση γεωγραφικού στίγματος στο χάρτη.
- Ένδειξη ποιότητας σήματος GPS.
- Αναπαραγωγή μουσικής στο παρασκήνιο κατά τη χρήση του ταχύμετρου.
- Εμφάνιση συνθηκών καιρού για την τρέχουσα τοποθεσία.



Εικόνα 10 Στιγμιότυπα από την εφαρμογή «GPS Speedometer» (Guru Info Media).

Η εφαρμογή «GPS Speedometer» (Guru Info Media) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

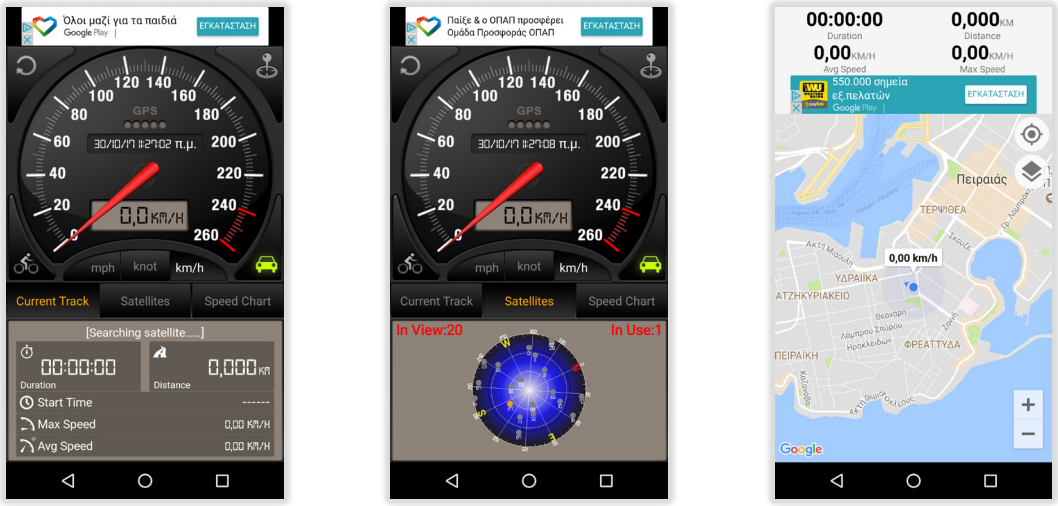
https://play.google.com/store/apps/details?id=com.guruinfomedia.gps.speedometer

1.1.7 Speedometer GPS (luozirui)

Η εφαρμογή «Speedometer GPS» προσφέρεται από τον προγραμματιστή εφαρμογών Android, luozirui, ο οποίος δραστηριοποιείται στο κομμάτι του προγραμματισμού από το 2012.

❖ Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:

- Ένδειξη της τρέχουσας ταχύτητας σε ψηφιακό και αναλογικό ταχύμετρο.
- Παρακολούθηση διανυθείσας απόστασης, χρόνου, μέγιστης και μέσης ταχύτητας.
- Ένδειξη υψόμετρου.
- Εναλλαγή μεταξύ μονάδων μέτρησης (mph, κόμβος ή km/h).
- Επιλογή τύπου ταξιδιού (αυτοκίνητο, ποδήλατο).
- Εμφάνιση γεωγραφικού στίγματος στο χάρτη.
- Ένδειξη ποιότητας σήματος GPS.
- Αποθήκευση διαδρομής σε αρχείο gpx.



Εικόνα 11 Στιγμιότυπα από την εφαρμογή «Speedometer GPS» (luozirui).

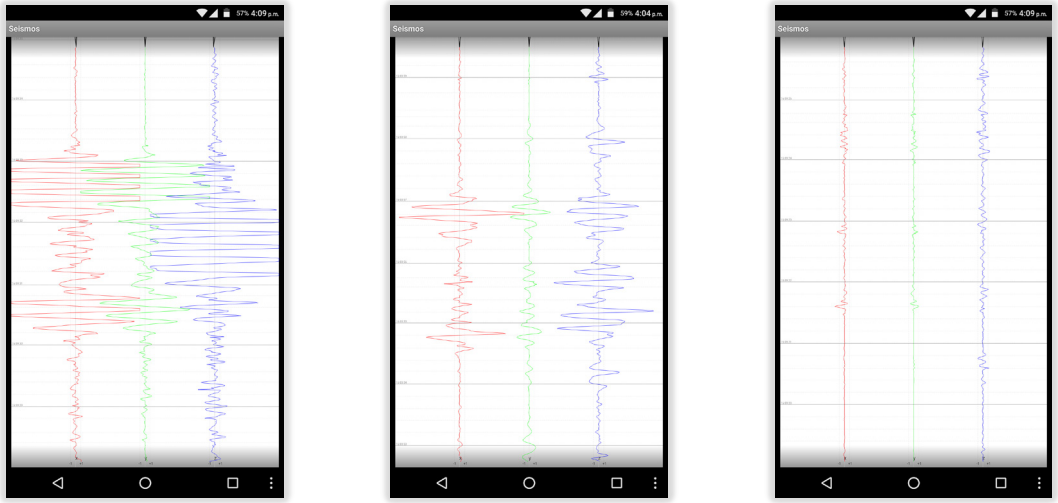
Η εφαρμογή «Speedometer GPS» (luozirui) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

https://play.google.com/store/apps/details?id=luo.speedometergps

1.2 Εφαρμογές Εσωτερικού

1.2.1 Seismos (Panayiotis Mavrommatis)

Η εφαρμογή «Seismos» προσφέρεται από τον Έλληνα προγραμματιστή Παναγιώτη Μαυρομμάτη, η οποία προβάλλει με γραφικό τρόπο τις αναταράξεις της συσκευής, που προκαλούνται κατά τη διάρκεια μιας σεισμικής δόνησης. *Αντίστοιχα, θα μπορούσε να χρησιμοποιηθεί για τη μέτρηση του μεγέθους των κακοτεχνιών στους δρόμους.*



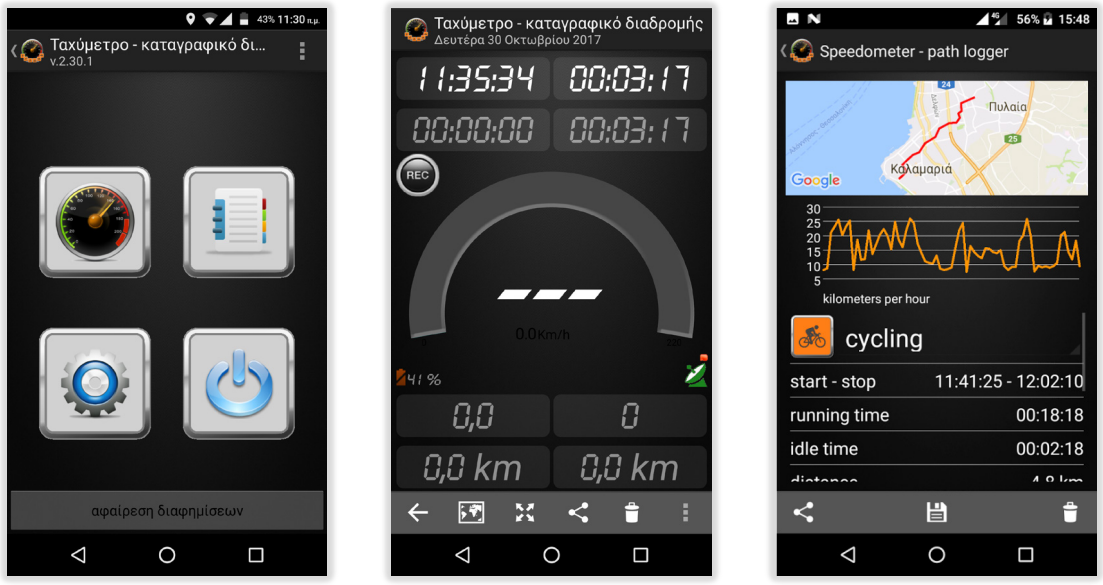
Εικόνα 12 Στιγμιότυπα από την εφαρμογή «Seismos» (Panayiotis Mavrommatis).

Η εφαρμογή «Seismos» (Panayiotis Mavrommatis) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:
https://play.google.com/store/apps/details?id=com.isovitis

1.2.2 Ταχύμετρο - καταγραφικό (Christos Themelis)

Η εφαρμογή «Ταχύμετρο - καταγραφικό» προσφέρεται από τον Έλληνα προγραμματιστή Χρήστο Θέμελη, ο οποίος εδρεύει στη Θεσσαλονίκη.

- Κύρια χαρακτηριστικά της εφαρμογής είναι τα ακόλουθα:
  - Λειτουργεί αποκλειστικά με χρήση GPS και όχι μέσω των δικτύων της συσκευής.
  - Ένδειξη της τρέχουσας ταχύτητας.
  - Παρακολούθηση διανυθείσας απόστασης, χρόνου, μέγιστης και μέσης ταχύτητας.
  - Ένδειξη ποιότητας σήματος GPS.
  - Εναλλαγή μεταξύ μονάδων μέτρησης (m/s, ft/s, km/h, m/h, κόμβοι).
  - Εμφάνιση ορίων ταχύτητας.
  - Προβολή τρέχουσας ώρας.
  - Λειτουργία νυκτός.
  - Υπολογιστής ταξιδιού.
  - Πολύγλωσση υποστήριξη.



Εικόνα 13 Στιγμιότυπα από την εφαρμογή «Ταχύμετρο - καταγραφικό» (Christos Themelis).

Η εφαρμογή «Ταχύμετρο - καταγραφικό» (Christos Themelis) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:
https://play.google.com/store/apps/details?id=com.themelisx.myspeedometer

ΚΕΦΑΛΑΙΟ 2: Αρχιτεκτονική Συστήματος

2.1 Πλατφόρμα Υλοποίησης

Η ανάπτυξη της εφαρμογής έγινε στην πλατφόρμα **Android**, χρησιμοποιώντας την τελευταία έκδοση του ολοκληρωμένου προγραμματιστικού περιβάλλοντος (IDE) **Android Studio 3.0**.

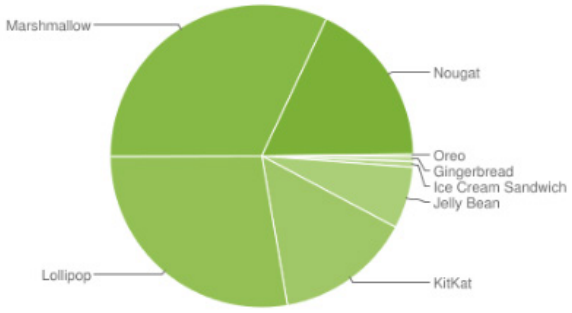
Το Android Studio είναι βασισμένο στο λογισμικό της JetBrains' IntelliJ IDEA, είναι σχεδιασμένο αποκλειστικά για προγραμματισμό Android (Ducrohet & Norbye & Chou, 2013) και αποτελεί το κύριο ολοκληρωμένο προγραμματιστικό περιβάλλον της Google για ανάπτυξη εφαρμογών Android. Είναι διαθέσιμο για Windows, Linux και Mac OS X και διατίθεται ελεύθερα προς χρήση.

Η γλώσσα ανάπτυξης που χρησιμοποιήθηκε είναι **Java**, αντικειμενοστρεφής γλώσσα προγραμματισμού, που σχεδιάστηκε από την εταιρία πληροφορικής Sun Microsystems και πρωτοεμφανίστηκε πριν από 22 χρόνια, στις 23 Μαΐου του 1995 (Binstock, 2015).

2.2 Συχνότητα Κατανομής Εκδόσεων Android

Στην εικόνα που ακολουθεί παρουσιάζονται στατιστικά στοιχεία σχετικά με τον αριθμό των συσκευών που τρέχουν μία συγκεκριμένη έκδοση της πλατφόρμας Android, όπως αυτά παρουσιάζονται στο επίσημο site των προγραμματιστών Android ("Platform Versions", 2017). Τα στοιχεία αφορούν χρονικό διάστημα 7 ημερών, με καταληκτική ημερομηνία τις 2 Οκτωβρίου 2017. Εκδόσεις Android με συχνότητα κατανομής μικρότερη από 0,1% δεν εμφανίζονται.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.3%
4.2.x		17	3.3%
4.3		18	1.0%
4.4	KitKat	19	14.5%
5.0	Lollipop	21	6.7%
5.1		22	21.0%
6.0	Marshmallow	23	32.0%
7.0	Nougat	24	15.8%
7.1		25	2.0%
8.0	Oreo	26	0.2%



Εικόνα 14 Συχνότητα κατανομής των εκδόσεων Android.

Παρατηρούμε λοιπόν πως συσκευές με λειτουργικό σύστημα προγενέστερο του Ice Cream Sandwich κοντεύουν να εκλείψουν. Για το λόγο αυτό, η παρούσα εφαρμογή, προσπαθώντας να καλύψει μία μεγάλη γκάμα συσκευών, **στοχεύει σε εφαρμογές από Ice Cream Sandwich και πάνω (Επίπεδο API 15 - Android 4.0.3 και Android 4.0.4)**. Ο περιορισμός αυτός δηλώνεται στο Gradle Script της εφαρμογής, με το χαρακτηριστικό «`minSdkVersion`».

Platform Version	API Level	VERSION_CODE	Notes
Android O	26	O	Platform Highlights
Android 7.1.1	25	N_MR1	Platform Highlights
Android 7.1			
Android 7.0	24	N	Platform Highlights
Android 6.0	23	M	Platform Highlights
Android 5.1	22	LOLLIPOP_MR1	Platform Highlights
Android 5.0	21	LOLLIPOP	
Android 4.4W	20	KITKAT_WATCH	KitKat for Wearables Only
Android 4.4	19	KITKAT	Platform Highlights
Android 4.3	18	JELLY_BEAN_MR2	Platform Highlights
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1	Platform Highlights
Android 4.1, 4.1.1	16	JELLY_BEAN	Platform Highlights
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1	Platform Highlights
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH	
Android 3.2	13	HONEYCOMB_MR2	
Android 3.1.x	12	HONEYCOMB_MR1	Platform Highlights
Android 3.0.x	11	HONEYCOMB	Platform Highlights
Android 2.3.4	10	GINGERBREAD_MR1	Platform Highlights
Android 2.3.3			
Android 2.3.2	9	GINGERBREAD	
Android 2.3.1			
Android 2.3			
Android 2.2.x	8	FROYO	Platform Highlights
Android 2.1.x	7	ECLAIR_MR1	Platform Highlights
Android 2.0.1	6	ECLAIR_0_1	
Android 2.0	5	ECLAIR	
Android 1.6	4	DONUT	Platform Highlights
Android 1.5	3	CUPCAKE	Platform Highlights
Android 1.1	2	BASE_1_1	
Android 1.0	1	BASE	

Πίνακας 1 Το επίπεδο API που υποστηρίζεται από κάθε έκδοση της πλατφόρμας Android (“What is API Level”, 2017).

2.3 Ενημερώσεις Τοποθεσίας

Μεγάλο μέρος των εφαρμογών που αναπτύσσονται για κινητές συσκευές, στηρίζονται στη συλλογή δεδομένων που αφορούν την τοποθεσία των χρηστών. Επομένως, συχνά προκύπτουν ερωτήματα σχετικά με τις μεθόδους που πρέπει να χρησιμοποιηθούν, ούτως ώστε να επιτευχθούν ακριβέστερα αποτελέσματα, χωρίς όμως να γίνεται μεγάλη κατανάλωση της μπαταρίας των συσκευών αυτών.

Για την απόκτηση της τρέχουσας τοποθεσίας του χρήστη στο Android μπορούμε είτε να χρησιμοποιήσουμε το **Location API** που είναι μέρος του Android framework, είτε να χρησιμοποιήσουμε το **API Υπηρεσιών Τοποθεσίας Google (Google Location Services API)**, το οποίο αποτελεί μέρος των Υπηρεσιών Google Play (Echessa, 2015).

2.3.1 Android Location API

Το **Location API** είναι μέρος του Android framework από το API επιπέδου 1 και μέσω της τάξης «`LocationManager`» παρέχεται η πρόσβαση στις υπηρεσίες τοποθεσίας του συστήματος. Οι υπηρεσίες αυτές επιτρέπουν στις εφαρμογές να λαμβάνουν περιοδικές ενημερώσεις της γεωγραφικής θέσης της συσκευής ή να στέλνουν μία Πρόθεση (Intent) σε άλλες εφαρμογές, με σκοπό να τις ενημερώσουν πως η συσκευή βρίσκεται κοντά στην προκαθορισμένη τοποθεσία.

Χρησιμοποιεί τρεις διαφορετικούς παρόχους (providers) για να λάβει την τοποθεσία, οι οποίοι ορίζονται προγραμματιστικά:

- **LocationManager.GPS_PROVIDER**: Ο πάροχος αυτός προσδιορίζει την τοποθεσία χρησιμοποιώντας δορυφόρους. Ανάλογα με τις συνθήκες, ενδέχεται να χρειαστεί λίγο χρόνο για να επιστρέψει μία σταθεροποιημένη θέση (location fix).
- **LocationManager.NETWORK_PROVIDER**: Ο πάροχος αυτός προσδιορίζει την τοποθεσία βάσει της διαθεσιμότητας του δικτύου τηλεφωνίας και σημείων πρόσβασης WiFi.
- **LocationManager.PASSIVE_PROVIDER**: Ο πάροχος αυτός επιστρέφει τοποθεσίες που δημιουργούνται από άλλους παρόχους. Οι ενημερώσεις τοποθεσίες λαμβάνονται παθητικά, όταν άλλες εφαρμογές ή υπηρεσίες ζητήσουν ενημερώσεις τοποθεσίας.

Για τη λήψη των ενημερώσεων τοποθεσίας μέσω του Location API του Android, γίνεται χρήση του πακέτου «`android.location.LocationListener`».

2.3.2 Google Location Services API

Το **API Υπηρεσιών Τοποθεσίας Google (Google Location Services API)**, μέρος των υπηρεσιών του Google Play, παρέχει ένα ισχυρότερο framework υψηλού επιπέδου, που αυτοματοποιεί τις εργασίες, όπως την επιλογή του παροχέα τοποθεσίας (GPS ή Networks) και τη διαχείριση ενέργειας. Η Ανίχνευση Δραστηριότητας είναι νέα λειτουργία που έχει προστεθεί στις Υπηρεσίες Τοποθεσίας και μπορεί να ανιχνεύσει εάν ο χρήστης κινείται με όχημα ή ποδήλατο, εάν είναι πεζός και τρέχει ή περπατάει ή εάν είναι ακίνητος. Δε δίνει συγκεκριμένα δεδομένα, αλλά μόνο την πιθανότητα να συμβεί κάποια δραστηριότητα. Εναπόκειται στην κρίση του προγραμματιστή ο τρόπος που θα διαχειριστεί τα δεδομένα αυτά.

Χρησιμοποιεί ένα πάροχο (provider) για να λάβει την τοποθεσία, που ονομάζεται **Fused Location Provider**. Ο πάροχος αυτός επιλέγει αυτόματα ποιον υποκείμενο πάροχο θα χρησιμοποιήσει, με βάση την ακρίβεια, τη χρήση μπαταρίας και άλλες παραμέτρους που του ορίζουμε προγραμματιστικά. Έχει άμεση πρόσβαση στη βέλτιστη και στην τελευταία γνωστή

τοποθεσία της συσκευής. Επιπλέον, έχει υψηλότερη ακρίβεια και χαμηλότερη κατανάλωση της μπαταρίας. Η αλληλεπίδραση με τον Fused Location Provider μπορεί να επιτευχθεί με δύο τρόπους:

1) Μέσω του interface **FusedLocationProviderApi**: Η πρόσβαση στον πάροχο γίνεται μέσω του *GoogleApiClient API της Google*.
2) Μέσω της τάξης **FusedLocationProviderClient**: Η πρόσβαση στον πάροχο γίνεται μέσω του *LocationServices API της Google*.

Σύμφωνα με το Μηχανικό Λογισμικού των Google Play Services, Stacy (2017), ο δεύτερος τρόπος είναι και ο βέλτιστος, δηλαδή μέσω του *LocationServices API*, καθώς:

- Οι κλήσεις API περιμένουν αυτόματα την υπηρεσία να συνδεθεί, χωρίς να υπάρχει η ανάγκη να περιμένουμε πριν να υποβάλλουμε κάποιο αίτημα, σε αντίθεση με το GoogleApiClient, το οποίο θα πρέπει να συνδεθεί πριν να εκτελέσουμε οποιαδήποτε ενέργεια.
- Χρησιμοποιεί το «`Task`» API της Google, το οποίο διευκολύνει τη σύνταξη ασύγχρονων εργασιών, σε αντίθεση με το GoogleApiClient, το οποίο χρησιμοποιεί το «`PendingResult`» API της Google.
- Ο κώδικας είναι αυτοτελής και μπορεί εύκολα να μετακινηθεί σε μία κοινόχρηστη τάξη.

Για τη λήψη των ενημερώσεων τοποθεσίας μέσω του Location Services API της Google, γίνεται χρήση του πακέτου «`com.google.android.gms.location.LocationCallback`».

Σύμφωνα με την επίσημη τεκμηρίωση (“Package android.location”, 2017), το Location API, που είναι μέρος του Android framework, δε συστήνεται πλέον για την πρόσβαση στην τοποθεσία των συσκευών Android. Ο προτιμώμενος τρόπος για την προσθήκη χωρικής επίγνωσης (location-awareness) στην εφαρμογή είναι η χρήση του API Υπηρεσιών Τοποθεσίας Google, διότι προσφέρει ένα απλούστερο API, υψηλότερη ακρίβεια και χαμηλότερη κατανάλωση.

Να σημειωθεί πως σε συσκευές όπου οι Υπηρεσίες Google δεν είναι διαθέσιμες, η χρήση του Location API του Android framework είναι μονόδρομος.

2.4 Βιβλιοθήκες Εφαρμογής

Στις εξαρτήσεις του Gradle Script της εφαρμογής, μεταξύ άλλων, ορίζουμε τις βιβλιοθήκες που απαιτούνται για τις ανάγκες της εκάστοτε εφαρμογής.

Στην παρούσα εφαρμογή οι βιβλιοθήκες που απαιτούνται είναι οι παρακάτω, οι οποίες θα αναλυθούν στις ενότητες που ακολουθούν:

- ✓ **Βιβλιοθήκες Υποστήριξης Android**
  - Βιβλιοθήκη appcompat v7
  - Βιβλιοθήκη Υποστήριξης Σχεδιασμού
  - Βιβλιοθήκη Υποστήριξης ConstraintLayout
- ✓ **Google Play Services**
  - Google Account Login API
  - Google Location and Activity Recognition API
  - Google Maps API
  - Google Maps Android API Utility Library

- ✓ **Firebase**
  - Firebase Realtime Database
  - Firebase Authentication
  - Firebase Crash Reporting
- ✓ **Βιβλιοθήκες Android Τρίτων**
  - Βιβλιοθήκη Picasso
  - Βιβλιοθήκη RoundedImageView
  - Βιβλιοθήκη GaugeView

2.5 Βιβλιοθήκες Υποστήριξης Android

Οι Βιβλιοθήκες Υποστήριξης Android (Android Support Libraries) είναι ένα σύνολο βιβλιοθηκών, που παρέχουν εκδόσεις των Android framework APIs, οι οποίες είναι **προς τα πίσω συμβατές (backward compatible)**. Κάθε Βιβλιοθήκη Υποστήριξης είναι προς τα πίσω συμβατή με ένα συγκεκριμένο επίπεδο Android API. Συνεπώς, κινητές συσκευές οι οποίες τρέχουν προηγούμενες εκδόσεις του Android, μπορούν να επωφελούνται από τα νεότερα APIs.

2.5.1 Βιβλιοθήκη appcompat v7

Η **βιβλιοθήκη appcompat v7 (v7 appcompat library)** συχνά αναφέρεται ως AppCompat και είναι μέρος των Βιβλιοθηκών Υποστήριξης v7 (v7 Support Libraries). Η βιβλιοθήκη αυτή παρέχει υλοποιήσεις για το ActionBar (εισήχθη στο API 11) και το ToolBar (εισήχθη στο API 21) που υποστηρίζονται προς τα πίσω έως και το API 7. Εξαρτάται από τη Βιβλιοθήκη Υποστήριξης v4, συνεπώς οποιοδήποτε χαρακτηριστικό εξαρτάται από το appcompat-v7, εξαρτάται επίσης και από το appcompat-v4.

Για να συμπεριλάβουμε τη βιβλιοθήκη appcompat v7, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής, το οποίο δηλώνει μία εξάρτηση στην έκδοση `27.0.0` της βιβλιοθήκης «`appcompat-v7`», που βρίσκεται μέσα στο πακέτο (namespace) «`com.android.support`»:

```
implementation 'com.android.support:appcompat-v7:27.0.0'
```

2.5.2 Βιβλιοθήκη Υποστήριξης Σχεδιασμού

Η **Βιβλιοθήκη Υποστήριξης Σχεδιασμού (Design Support Library)** παρέχει APIs για να υποστηρίξει την προσθήκη συστατικών σχεδιασμού υλικού (material design components) και μοτίβων (patterns) στις εφαρμογές (π.χ. navigation drawers, floating action buttons - FAB, snackbars και tabs).

Για να συμπεριλάβουμε τη Βιβλιοθήκη Υποστήριξης Σχεδιασμού, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.android.support:design:27.0.0'
```

2.5.3 Βιβλιοθήκη Υποστήριξης ConstraintLayout

Η **Βιβλιοθήκη Υποστήριξης ConstraintLayout** εισήχθη στο API 9 και συνίσταται από την Google ως το ιδανικότερο ViewGroup (δηλαδή ως το view-γονέα που μπορεί να περικλείει άλλα

views, τα παιδιά) για τη σχεδίαση περιβαλλόντων χρήστη που προσαρμόζονται δυναμικά και άμεσα σε κάθε προσανατολισμό και μέγεθος οθόνης (“Build a Responsive UI with ConstraintLayout”, 2017).

Το ConstraintLayout επιτρέπει τη δημιουργία μεγάλων και σύνθετων διατάξεων (layouts) με ιεραρχία επίπεδης προβολής (χωρίς ένθετες ομάδες προβολής). Όλες οι προβολές (views) σχεδιάζονται σύμφωνα με τις σχέσεις μεταξύ των αδελφών προβολών (sibling views) και της γονικής διάταξης (parent layout).

Για να συμπεριλάβουμε τη Βιβλιοθήκη Υποστήριξης ConstraintLayout, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

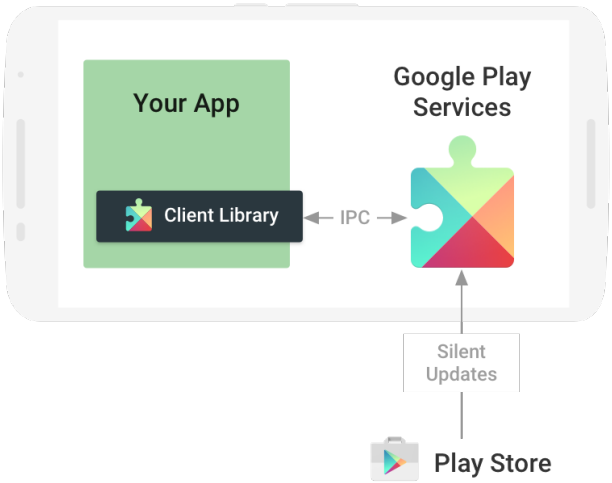
```
implementation 'com.android.support.constraint:constraint-
layout:1.0.2'
```

2.6 Google Play Services

Τα **Google Play Services** είναι υπηρεσίες της Google, μέσω των οποίων οι προγραμματιστές μπορούν εύκολα να ενσωματώσουν στις εφαρμογές τους τις νεότερες υπηρεσίες που υποστηρίζονται από την Google, όπως οι Χάρτες, το Google+, και άλλα (“Overview of Google Play Services”, 2017).

Η βιβλιοθήκη του πελάτη (**client**) περιέχει τις διασυνδέσεις στις μεμονωμένες υπηρεσίες της Google και επιτρέπει στους προγραμματιστές να λάβουν εξουσιοδότηση από τους χρήστες, ούτως ώστε να αποκτήσουν πρόσβαση στις υπηρεσίες αυτές με τα διαπιστευτήριά τους.

Το APK των υπηρεσιών του Google Play **(Google Play Services APK)** περιέχει τις μεμονωμένες υπηρεσίες της Google και εκτελείται ως υπηρεσία παρασκηνίου στο λειτουργικό σύστημα Android. Η αλληλεπίδραση με την υπηρεσία παρασκηνίου είναι εφικτή μέσω της βιβλιοθήκης του πελάτη (client library) και η υπηρεσία εκτελεί ενέργειες για λογαριασμό της εφαρμογής.



Εικόνα 15 Το APK των Υπηρεσιών Googleblog Play σε συσκευές χρηστών λαμβάνει τακτικές ενημερώσεις για νέα APIs, λειτουργίες και διορθώσεις σφαλμάτων.

Το APK των Υπηρεσιών Google Play είναι διαθέσιμο αποκλειστικά μέσω του **Google Play Store**. Οι χρήστες λαμβάνουν άμεσα τις ενημερώσεις για νέα APIs, λειτουργίες και διορθώσεις σφαλμάτων. Συσκευές με λειτουργικό προγενέστερο του Android 2.3 (API επιπέδου 9) ή συσκευές οι οποίες δε διαθέτουν το Google Play Store δεν υποστηρίζονται.

Στις εκδόσεις των Υπηρεσιών Google Play πριν από την έκδοση 6.5, ήταν αναγκαία η μεταγλώττιση ολόκληρου του πακέτου των APIs στην εφαρμογή, μέσω της ακόλουθης εξάρτησης:

```
implementation 'com.google.android.gms:play-services:11.4.2'
```

Από την έκδοση 6.5 και μετά υπάρχει η δυνατότητα μεταγλώττισης επιλεκτικών APIs. Στην παρούσα εφαρμογή έχουν συμπεριληφθεί τα ακόλουθα APIs των Υπηρεσιών Google Play:

- Google Account Login API.
- Google Location and Activity Recognition API.
- Google Maps API.

2.6.1 Google Account Login API

Η πιστοποίηση των χρηστών μέσω του API Google Account Login (“Google Sign-In”) είναι ένα ασφαλές σύστημα ελέγχου ταυτότητας, επιτρέποντας στους χρήστες να συνδεθούν στην εφαρμογή με τον Google λογαριασμό τους, τον ίδιο λογαριασμό δηλαδή που χρησιμοποιούν ήδη στο Play Store, στο Google+ και σε άλλες υπηρεσίες της Google.

Για να συμπεριλάβουμε το API Google Account Login, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.android.gms:play-services-auth:11.4.2'
```

Στην παρούσα εφαρμογή, η πιστοποίηση των χρηστών υλοποιείται συνδυάζοντας τη μέθοδο Google Sign-In, με την υπηρεσία πιστοποίησης που παρέχει η Firebase, όπως αυτή θα εξηγηθεί σε επόμενη ενότητα.

2.6.2 Google Location and Activity Recognition API

Για την απόκτηση της τρέχουσας τοποθεσίας του χρήστη στο Android, όπως προαναφέρθηκε στην παράγραφο 2.3.2, επιλέγουμε το **API Υπηρεσιών Τοποθεσίας Google (Google Location Services API)**, το οποίο αποτελεί μέρος των Υπηρεσιών Google Play.

Εικόνα 16 Υπηρεσίες τοποθεσίας.

Το API Υπηρεσιών Τοποθεσίας Google παρέχει ένα ισχυρό framework υψηλού επιπέδου, με αυτοματοποιημένες εργασίες που αφορούν τον παροχέα τοποθεσίας (GPS ή Networks) και τη διαχείριση ενέργειας. Η Ανίχνευση Δραστηριότητας μπορεί να ανιχνεύσει την πιθανότητα ο χρήστης να κινείται με όχημα ή ποδήλατο, εάν είναι πεζός και τρέχει ή περπατάει ή εάν είναι ακίνητος.

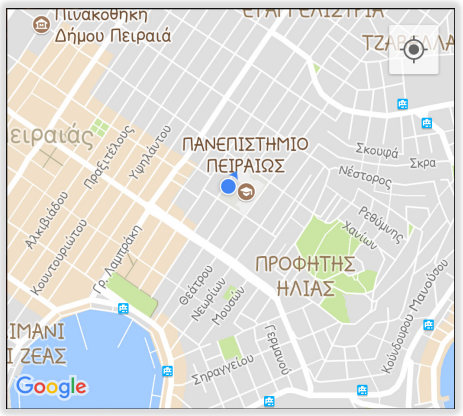
Εικόνα 17 Αναγνώριση της δραστηριότητας του χρήστη.

Για να συμπεριλάβουμε το API Google Location and Activity Recognition, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.android.gms:play-services-location:11.4.2'
```

2.6.3 Google Maps API

Η ενσωμάτωση και διαχείριση των χαρτών της Google σε εφαρμογές Android, γίνεται μέσω του API Google Maps. Χειρίζεται αυτόματα την πρόσβαση στους διακομιστές των χαρτών Google, τη λήψη δεδομένων, την απεικόνιση του χάρτη και την απόκριση στις χειρονομίες (gestures) πάνω στο χάρτη.



Εικόνα 18 Χάρτης Google σε εφαρμογή Android.

Μέσω του API Google Maps μπορούμε να προσθέσουμε τα ακόλουθα γραφικά πάνω σε ένα χάρτη της Google (“Introduction to the Google Maps Android API”, 2017):

- Εικονίδια αγκυροβολημένα σε συγκεκριμένες θέσεις στο χάρτη (Markers).
- Σετ από τμήματα γραμμών (Polylines).
- Εγκλεισμένα τμήματα (Polygons).

- Γραφικά bitmap αγκυροβολημένα σε συγκεκριμένες θέσεις στο χάρτη (Ground Overlays).
- Σετ εικόνων που εμφανίζονται πάνω από τα κύρια πλακίδια του χάρτη (Tile Overlays).

Για να συμπεριλάβουμε το API Google Maps, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.android.gms:play-services-map:11.4.2'
```

2.6.4 Google Maps Android API Utility Library

Η βιβλιοθήκη ανοιχτού κώδικα Google Maps Android API Utility Library περιέχει βοηθητικά εργαλεία για την προσθήκη προηγμένων λειτουργιών στους χάρτες της Google. Οι λειτουργίες αυτές είναι:

- Εισαγωγή GeoJSON στο χάρτη.
- Εισαγωγή KML στο χάρτη.
- Εισαγωγή Heatmaps στο χάρτη.
- Προσαρμογή των αγκυροβολημένων εικονιδίων (markers) μέσω εικονιδίων φούσκας (bubble icons).
- Διαχείριση ομάδων από δείκτες.
- Κωδικοποίηση και αποκωδικοποίηση σετ, αποτελούμενων από τμήματα γραμμών (polylines).
- Υπολογισμός αποστάσεων, περιοχών και επικεφαλίδων μέσω σφαιρικής γεωμετρίας.

Για να συμπεριλάβουμε τη βιβλιοθήκη αυτή, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.maps.android:android-maps-utils:0.5+'
```

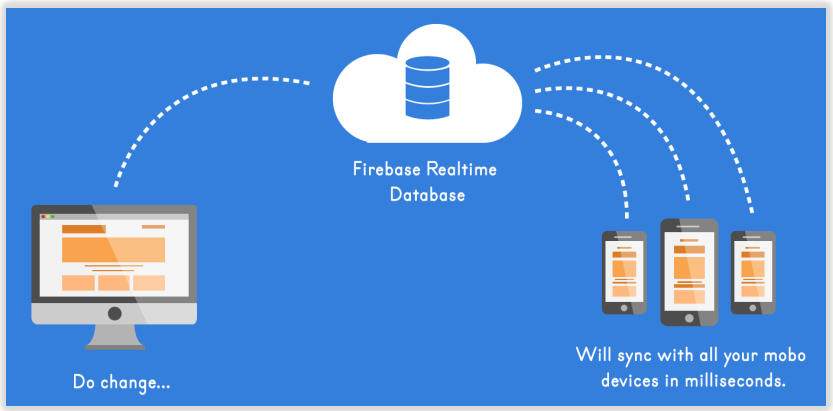
2.7 Firebase

Η **Firebase** είναι η διαδικτυακή πλατφόρμα της Google, που παρέχει εργαλεία για την ανάπτυξη mobile και web εφαρμογών υψηλής ποιότητας. Οι βασικές ανάγκες της εφαρμογής καλύπτονται από το δωρεάν πλάνο της Firebase, που ονομάζεται **Spark** (“Firebase Pricing Plans”, 2017).

Εικόνα 19 Firebase, η διαδικτυακή πλατφόρμα της Google.

2.7.1 Firebase Realtime Database

Η Firebase Realtime Database είναι μία **NoSQL** βάση δεδομένων που φιλοξενείται στο **Σύννεφο (Cloud)**. Συνεπώς, τα δεδομένα παραμένουν διαθέσιμα στο σύννεφο, ακόμα και όταν μία εφαρμογή τεθεί εκτός σύνδεσης. Τα δεδομένα αποθηκεύονται σε μορφή JSON και συγχρονίζονται σε όλους τους πελάτες σε πραγματικό χρόνο.



Εικόνα 20 Firebase Realtime Database.

Για να μπορέσουμε να χρησιμοποιήσουμε τη Firebase Database, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.firebase:firebase-database:11.4.2'
```

Μία σημαντική λειτουργία που υποστηρίζει η Firebase είναι η **λειτουργία εκτός σύνδεσης**. Η Firebase χειρίζεται αυτόματα τις προσωρινές διακοπές του δικτύου. Τα προσωρινά αποθηκευμένα δεδομένα (cached data) είναι διαθέσιμα ενώ η εφαρμογή είναι εκτός σύνδεσης και η Firebase αποστέλλει εκ νέου κάθε εγγραφή όταν αποκαθίσταται η συνδεσιμότητα του δικτύου.

Η λειτουργία εκτός σύνδεσης ονομάζεται «**Disk Persistence**». Όταν η λειτουργία αυτή είναι ενεργοποιημένη, τότε η εφαρμογή γράφει τα δεδομένα τοπικά στη συσκευή, ούτως ώστε να είναι διαθέσιμα σε κατάσταση εκτός σύνδεσης, ακόμα και όταν ο χρήστης κάνει επανεκκίνηση της εφαρμογής ή και της συσκευής του.

Το στιγμιότυπο της τάξης `FirebaseDatabase` είναι το σημείο εισόδου για την πρόσβασή μας στη βάση δεδομένων της Firebase και η λειτουργία εκτός σύνδεσης ενεργοποιείται θέτοντας την παράμετρο **`true`** στη μέθοδο `setPersistenceEnabled()`.

```
FirebaseDatabase mFirebaseDB = FirebaseDatabase.getInstance();
mFirebaseDB.setPersistenceEnabled(true);
```

Το στιγμιότυπο της τάξης `DatabaseReference` αντιπροσωπεύει μία συγκεκριμένη τοποθεσία στη βάση δεδομένων της Firebase και μπορεί να χρησιμοποιηθεί για την ανάγνωση ή την εγγραφή δεδομένων στην τοποθεσία αυτή.

```
DatabaseReference fireDBRefStoreData =
                mFirebaseDB.getReference(MyConstants.KEY_FIREBASE_DATA);
```

Οι τύποι δεδομένων που μπορούμε να αποθηκεύσουμε στη βάση δεδομένων Firebase είναι:

- String
- Long
- Double
- Boolean
- Map<String, Object>
- List<Object>

Χρησιμοποιώντας τη μέθοδο «`setValue()`» αποθηκεύουμε ένα σετ δεδομένων στο μονοπάτι που ορίζουμε. Εάν το μονοπάτι αυτό δεν υπάρχει, τότε δημιουργούνται οι αντίστοιχοι κόμβοι που λείπουν, διαφορετικά το σετ δεδομένων προστίθεται κάτω από τα ήδη υπάρχοντα σετ του μονοπατιού αυτού.

Στην παρούσα εφαρμογή, το μονοπάτι που αποθηκεύουμε το σετ δεδομένων είναι το «`/data/$UserUID/$date/$timestamp`», όπου το σύμβολο «$» αναπαριστά τις μεταβλητές.

- `$UserUID` : Το ID του χρήστη.
- `$date` : Η ημερομηνία σε μορφή «yyyy-MM-dd».
- `$timestamp` : Χρονοσφραγίδα.

```
fireDBRefStoreData
    .child(mSharedPref.getString(MyConstants.SHARED_GOOGLE_UID, ""))
    .child(timestampToDate(timestampLastUpload))
    .child(Springvale(timestampLastUpload))
    .set Value(getNewDataAsMap(), new
                              DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(DatabaseError databaseError,
                               DatabaseReference databaseReference) {
            // If an error occurred, then...
            if (databaseError != null) {
                Log.e(LOG_TAG, " Data could not be saved "
                                          + databaseError.getMessage());
            }
            // Data saved successfully...
            else {
                Log.d(LOG_TAG, "Data saved successfully.");
            }
        }
    });
```

Σημείωση: Χρησιμοποιούμε το interface `DatabaseReference.CompletionListener`, ούτως ώστε να γνωρίζουμε εάν τα δεδομένα εισήχθησαν στη βάση επιτυχώς.

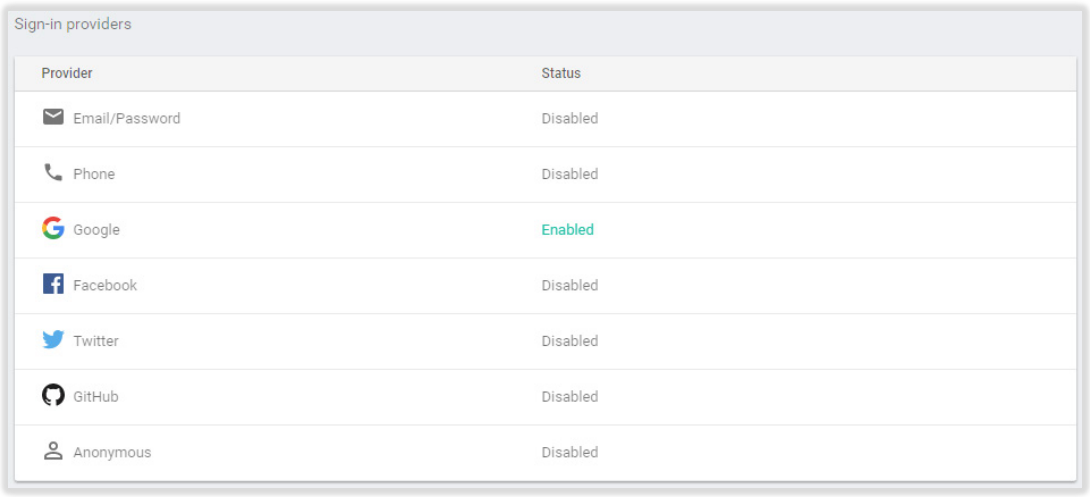
2.7.2 Firebase Authentication

Η υπηρεσία Firebase Authentication παρέχει υπηρεσίες backend και έτοιμες βιβλιοθήκες διεπαφής χρήστη (UI) για την πιστοποίηση των χρηστών στην εφαρμογή. Οι μέθοδοι πιστοποίησης που υποστηρίζει η Firebase είναι:

- **Password Authentication**: Χρήση διεύθυνσης email / κωδικού πρόσβασης.

- **Phone Number**: Χρήση τηλεφωνικού αριθμού, στον οποίο αποστέλλεται SMS με διαφορετικό κωδικό κάθε φορά.
- **Google Sign-In**: Χρήση λογαριασμού Google.
- **Facebook Login**: Χρήση λογαριασμού Facebook.
- **Twitter**: Χρήση λογαριασμού Twitter.
- **GitHub**: Χρήση λογαριασμού GitHub.
- **Anonymous Authentication**: Χρήση προσωρινού ανώνυμου λογαριασμού.

Στην παρούσα εφαρμογή η πιστοποίηση των χρηστών γίνεται αποκλειστικά μέσω της μεθόδου «Google Sign-In», δηλαδή χρησιμοποιείται το email που έχει ορίσει ο χρήστης για την είσοδό του στο Play Store.

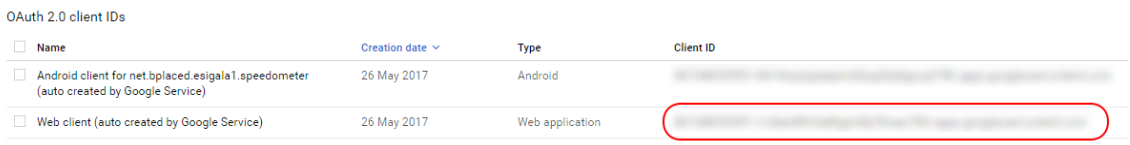


Εικόνα 21 Firebase Authentication: Επιλεγμένη μέθοδος «Google Sign-In».

Για να μπορέσουμε να χρησιμοποιήσουμε την υπηρεσία Firebase Authentication, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.firebase:firebase-auth:11.4.2'
```

Επίσης, θα χρειαστούμε το OAuth 2.0 Client ID του διακομιστή (backend server), το οποίο αντιστοιχεί στο κλειδί με όνομα «**Web client**» (*τύπου «Web application»*). Το κλειδί αυτό το βρίσκουμε στην καρτέλα «Credentials» της σελίδας «APIs & Services» της πλατφόρμας Google Cloud (https://console.cloud.google.com/apis/credentials):



Εικόνα 22 Το κλειδί «Web client» (Backend server's OAuth 2.0 client ID).

Ένα στιγμιότυπο της τάξης «`GoogleSignInOptions`» είναι απαραίτητο για να ενεργοποιήσουμε το Google Sign-In API. Δίνοντας την παράμετρο «`DEFAULT_SIGN_IN`»

ορίζουμε τις προεπιλεγμένες ρυθμίσεις για τη μέθοδο Google Sign-In, αποκτώντας πρόσβαση στο αναγνωριστικό του χρήστη (user's ID) και τα βασικά στοιχεία του προφίλ του. Επιπλέον, μέσω της μεθόδου «`requestEmail()`» ζητάμε τη διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη και το OAuth 2.0 Client ID του διακομιστή το περνάμε ως όρισμα στη μέθοδο «`requestIdToken()`»:

```
GoogleSignInOptions gso = new GoogleSignInOptions
        .Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(BuildConfig.AUTH_WEB_CLIENT_ID)
        .requestEmail()
        .build();
```

Επιπρόσθετα, χρειαζόμαστε ένα στιγμιότυπο της τάξης «`GoogleApiClient`» με πρόσβαση στο Google Sign-In API και τις ρυθμίσεις που ορίσαμε πιο πάνω. Καλώντας τη μέθοδο «`connect()`», συνδέουμε την εφαρμογή-πελάτη (client) με τις υπηρεσίες του Google Play:

```
final GoogleApiClient mGoogleApiClient = new GoogleApiClient
        .Builder(mContext)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();

mGoogleApiClient.connect();
```

Χρησιμοποιώντας το interface `GoogleApiClient.ConnectionCallbacks`, μπορούμε να γνωρίζουμε πότε η εφαρμογή-πελάτης (client) είναι συνδεδεμένη ή αποσυνδεδεμένη από την υπηρεσία. Όταν η εφαρμογή συνδεθεί επιτυχώς στην υπηρεσία Firebase Authentication, ξεκινάμε μία νέα Δραστηριότητα (Activity), περιμένοντας ένα αποτέλεσμα:

```
mGoogleApiClient.registerConnectionCallbacks(new
                                GoogleApiClient.ConnectionCallbacks() {
    @Override
    public void onConnected(@Nullable Bundle bundle) {
        if(mGoogleApiClient.isConnected()) {
            mActivity.startActivityForResult(Auth.GoogleSignInApi
            .getSignInIntent(mGoogleApiClient),
            REQUEST_CHECK_SIGN_IN);
            // Disconnect the Google API client.
            disconnectGoogleApiClient(mGoogleApiClient);
        }
    }
    @Override
    public void onConnectionSuspended(int i) {
        // Disconnect the Google API client.
        disconnectGoogleApiClient(mGoogleApiClient);
    }
});
```

Στη νέα Δραστηριότητα που προαναφέρθηκε, ο χρήστης καλείται να επιλέξει ένα λογαριασμό Google για να συνδεθεί στην εφαρμογή και το αποτέλεσμα της ενέργειας του χρήστη το λαμβάνουμε στη μέθοδο «`onActivityResult()`». Εάν ο χρήστης επιλέξει ένα λογαριασμό Google, τότε το αποτέλεσμα που λαμβάνουμε αντιστοιχεί στον κωδικό «***RESULT_OK***», και ενημερώνουμε κατάλληλα το UI της εφαρμογής με τα στοιχεία του πιστοποιημένου χρήστη, εκτελώντας τον απαραίτητο κώδικα. Σε αντίθετη περίπτωση, είτε εάν ο χρήστης αρνηθεί να

επιλέξει ένα λογαριασμό Google είτε εάν η εφαρμογή αποτύχει να πιστοποιήσει το χρήστη, τότε προβαίνουμε στην εκτέλεση του κατάλληλου κώδικα που αφορά στην αποτυχία πιστοποίησης του χρήστη:

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                                    Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // Result returned from launching the Google Sign In intent.
    if (requestCode == UserAuthorization.REQUEST_CHECK_SIGN_IN) {
        // RESULT_OK
        if (resultCode == RESULT_OK) {

            // Initialize the "mUserAuthorization" if needed.
            initializeUserAuthObject();

            // Handle the Google Sign In Result.
            mUserAuthorization.handleResultFromSignInIntent(data);

            // Set a new start time (duration on dashboard) and update
            // its value into the SharedPreferences object.
            setNewStartTimeIntoSharedPref();
        }
        // RESULT_CANCELED
        else {
            // Update the UI of the subclass.
            afterGoogleSignInCancellation();
        }
    }
    else {
        Log.d(LOG_TAG, "Irrelevant result code.");
     }
}
```

2.7.3 Firebase Crash Reporting

Η υπηρεσία Firebase Crash Reporting δημιουργεί λεπτομερείς αναφορές για τα σφάλματα της εφαρμογής, βοηθώντας τους προγραμματιστές να διαγνώσουν και να επιδιορθώσουν πιθανά προβλήματα των εφαρμογών που έχουν ήδη διαθέσει στο κοινό. Τα σφάλματα ομαδοποιούνται σε προβλήματα που βασίζονται σε παρόμοια ίχνη στοίβας (stack traces) και ταξινομούνται βάσει της σοβαρότητας των επιπτώσεων στους χρήστες της εφαρμογής.

Για να μπορέσουμε να χρησιμοποιήσουμε την υπηρεσία Firebase Crash Reporting, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.google.firebase: firebase-crash:11.4.2'
```

Στην παρούσα εφαρμογή γίνεται υλοποίηση του interface `Thread.UncaughtExceptionHandler`, το οποίο στέλνει αναφορές για τα σφάλματα που η εφαρμογή δεν μπορεί να διαχειριστεί μέσω των δηλώσεων «`try...catch`» και οδηγούν στην κατάρρευσή της (crash):

```
@Override
public void uncaughtException(Thread thread, Throwable throwable) {
    // Generate a crash report for the uncaught exception.
    FirebaseCrash.report(throwable);
}
```

2.8 Βιβλιοθήκες Android Τρίτων

Στο διαδίκτυο υπάρχουν αρκετές βιβλιοθήκες ανοιχτού κώδικα, γνωστές ως **Βιβλιοθήκες Android Τρίτων (Android Third-Party Libraries)**, κάθε μία εκ των οποίων εξυπηρετεί διαφορετικούς σκοπούς.

2.8.1 Βιβλιοθήκη Picasso

Η Βιβλιοθήκη ανοιχτού κώδικα **Picasso** είναι ιδιαίτερα δημοφιλής σε εφαρμογές Android, καθώς επιτρέπει την εύκολη φόρτωση και το μετασχηματισμό εικόνων στην εφαρμογή, με ελάχιστη χρήση της μνήμης (Picasso, 2013).

Για να συμπεριλάβουμε τη Βιβλιοθήκη Picasso, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.squareup.picasso:picasso:2.5.2'
```

2.8.2 Βιβλιοθήκη RoundedImageView

Η Βιβλιοθήκη Android ανοιχτού κώδικα **RoundedImageView** επιτρέπει τη δημιουργία στρογγυλεμένων γωνιών σε εικόνες.

Για να συμπεριλάβουμε τη Βιβλιοθήκη RoundedImageView, θα πρέπει να εισάγουμε την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation 'com.makeramen:roundedimageview:2.3.0'
```

2.8.3 Βιβλιοθήκη GaugeView

Στην παρούσα εφαρμογή, για τη σχεδίαση της βελόνας του ταχύμετρου (αναλογικό ταχύμετρο) έγινε χρήση της βιβλιοθήκης «`GaugeView`» (Sarajlija, 2015), η οποία παραμετροποιήθηκε για να καλύψει τις ανάγκες μας. Στη δομή της εφαρμογής εισάγεται ως έργο (project) με το όνομα «`gaugelibrarysig`» και με την ακόλουθη εξάρτηση στο Gradle Script της εφαρμογής:

```
implementation project(':gaugelibrarysig')
```

Επίσης, το όνομα του έργου αυτού θα πρέπει να προστεθεί στο Gradle Script των ρυθμίσεων της εφαρμογής, ούτως ώστε να συμπεριληφθεί κατά την κατασκευή (build) της:

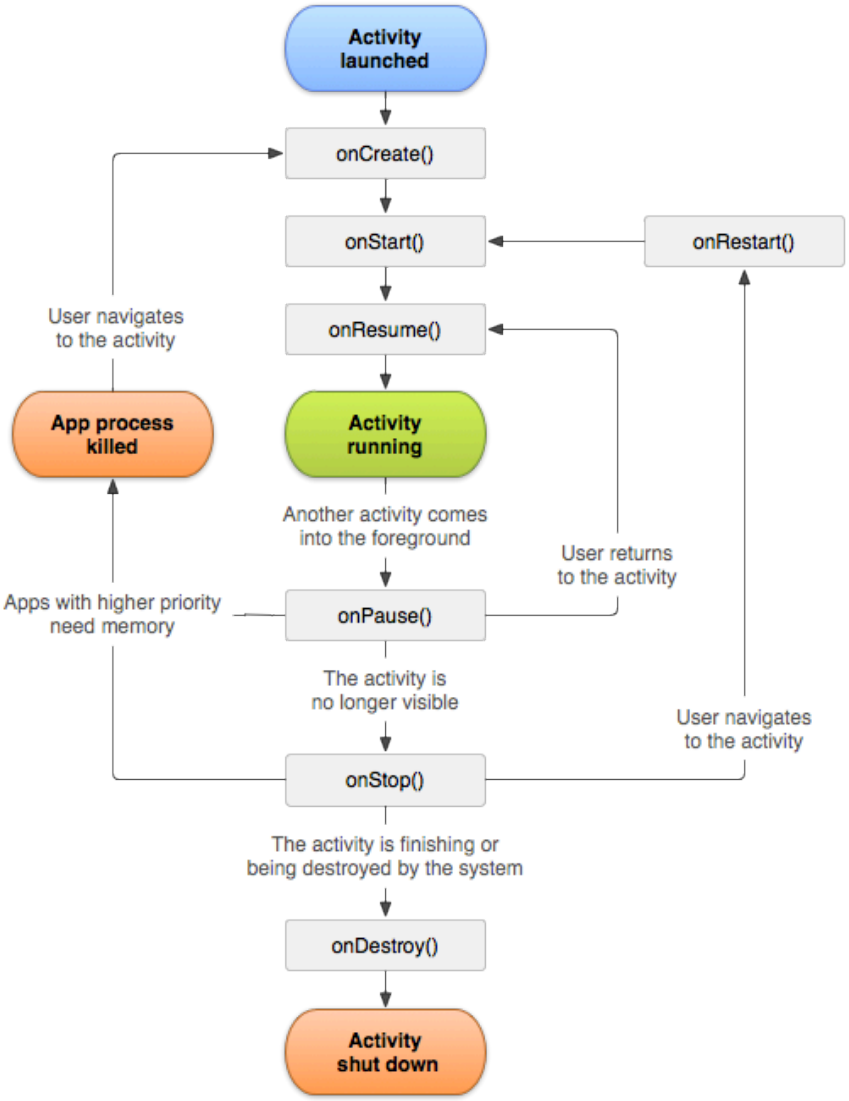
```
include ':gaugelibrarysig'
```

2.9 Κύκλος Ζωής Εξαρτημάτων

2.9.1 Activities

Οι Δραστηριότητες **(Activities)** είναι από τα θεμελιώδη δομικά στοιχεία των εφαρμογών στην πλατφόρμα Android. Χρησιμεύουν ως το σημείο εισόδου (entry point) για την αλληλεπίδραση ενός χρήστη με μια εφαρμογή, αλλά και για την περιήγηση του χρήστη μέσα στην εφαρμογή ή μεταξύ άλλων εφαρμογών (“Activities”, 2016). Αντιπροσωπεύει μία ενιαία οθόνη με το περιβάλλον του χρήστη (user interface). Η σωστή και εξειδικευμένη διαχείριση των δραστηριοτήτων διασφαλίζουν ότι:

- ✓ Οι αλλαγές προσανατολισμού πραγματοποιούνται ομαλά, προσφέροντας άριστη εμπειρία χρήστη, χωρίς διακοπές.
- ✓ Τα δεδομένα του χρήστη δεν χάνονται κατά τη μετάβαση από το ένα Activity στο άλλο.
- ✓ Το σύστημα καταστρέφει τις απαραίτητες διεργασίες, όταν κριθεί κατάλληλο.

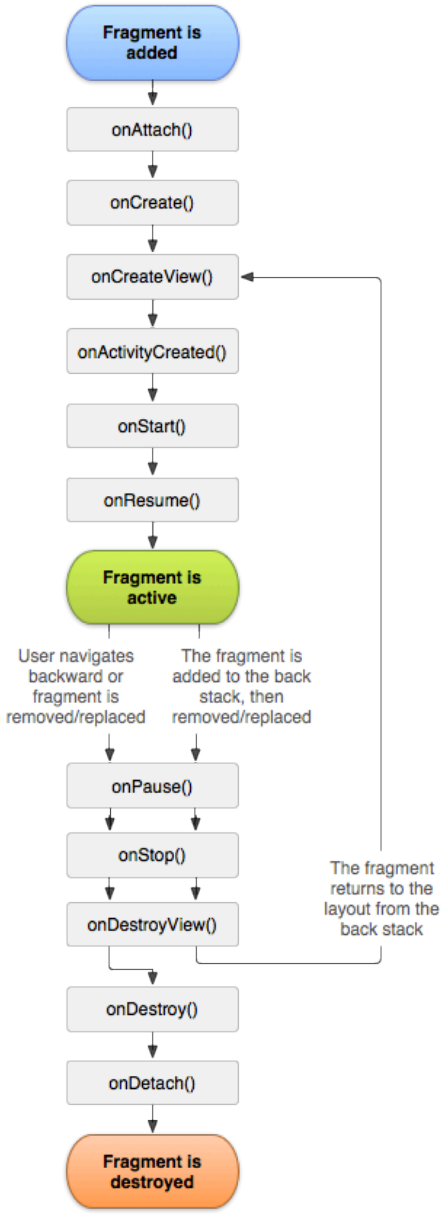


Εικόνα 23 Ο κύκλος ζωής ενός Activity.

2.9.2 Fragments

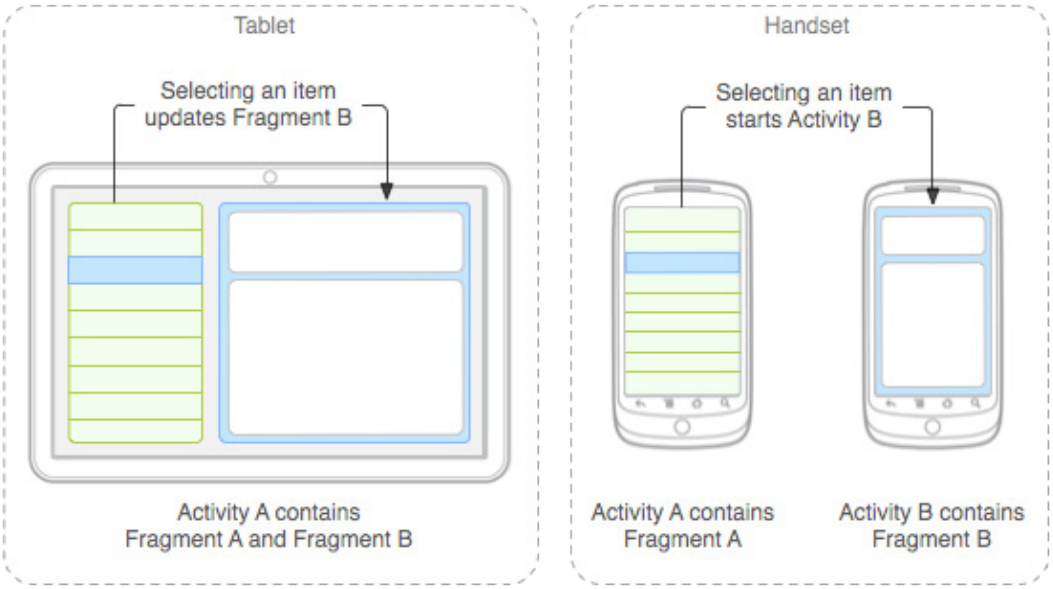
Τα **Τμήματα (Fragments)** είναι κομμάτια των Activities που επιτρέπουν μία δομοστοιχειωτή σχεδίαση, δηλαδή σχεδίαση βασισμένη σε ανεξάρτητες μονάδες. Κάθε Fragment έχει το δικό του κύκλο ζωής, λαμβάνει τα δικά του σήματα εισόδου και μπορεί να προστεθεί ή να αφαιρεθεί ενώ το Activity εκτελείται. Θα μπορούσε να θεωρηθεί ως Υπο-Δραστηριότητα (Sub Activity) που μπορεί να επαναχρησιμοποιηθεί σε διάφορα Activities. Υπάρχει η δυνατότητα να συνδυάσουμε πολλά Fragments σε ένα μόνο Activity για να δημιουργήσουμε ένα περιβάλλον χρήστη πολλαπλών παραθύρων (multi-pane UI), καθώς και να επαναχρησιμοποιήσουμε ένα Fragment σε πολλά Activities (“Fragments”, 2017).

Ο κύκλος ζωής των Fragments μοιάζει πολύ με τον κύκλο ζωής των Activities.



Εικόνα 24 Ο κύκλος ζωής ενός Fragment.

Ένα Fragment πρέπει πάντα να ενσωματώνεται σε ένα Activity και ο κύκλος ζωής του επηρεάζεται άμεσα από τον κύκλο ζωής του Activity που το φιλοξενεί.



Εικόνα 25 Δύο τμήματα UI που ορίζονται από Fragments, μπορούν να συνδυαστούν είτε σε ένα Activity για προβολή σε Tablets (αριστερά) είτε να ενσωματωθούν σε διαφορετικά Activities (ένα μόνο Fragment σε ένα Activity) για προβολή σε Smartphones (δεξιά).

2.9.3 Services

Οι **Υπηρεσίες (Services)** είναι εξαρτήματα των εφαρμογών Android και ανήκουν στα βασικά δομικά στοιχεία τους. Χρησιμεύουν ως σημεία εισόδου γενικού σκοπού, που μπορούν να εκτελούν εκτεταμένες εργασίες στο παρασκήνιο, χωρίς να παρέχουν διεπαφή με το χρήστη.

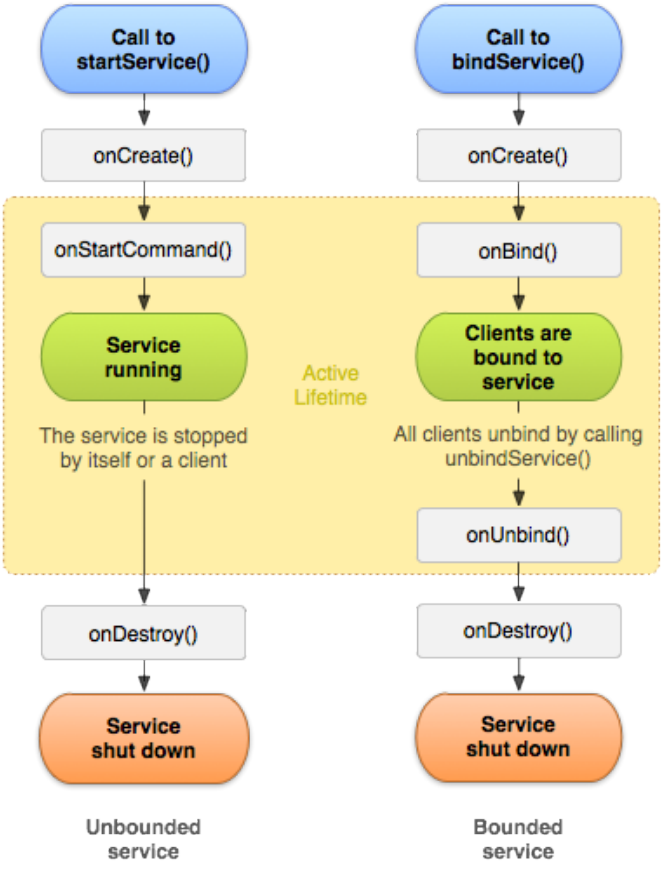
Ένα Service ξεκινάει να εκτελείται από κάποιο άλλο εξάρτημα της εφαρμογής και συνεχίζει να εκτελείται στο παρασκήνιο, ακόμα και αν ο χρήστης μεταβεί σε άλλη εφαρμογή. Επιπρόσθετα, κάποιο εξάρτημα μπορεί να συνδεθεί με ένα Service για να αλληλεπιδράσει μαζί του και να πραγματοποιήσει ακόμα και επικοινωνία μεταξύ των διεργασιών (interprocess communication - IPC). Για παράδειγμα, ένα Service μπορεί να χειριστεί συναλλαγές δεδομένων (network transactions), να παίξει μουσική, να κάνει εγγραφή σε αρχείο ή ανάγνωση από αυτό (I/O) ή να αλληλεπιδράσει με έναν παροχέα περιεχομένου (content provider), όλα από το παρασκήνιο (“Services”, 2017).

Τα Services χωρίζονται σε τρεις κατηγορίες:

- **Υπηρεσία Προσκηνίου (Foreground Service)**: Εκτελεί κάποια λειτουργία που είναι εμφανής στο χρήστη. Για παράδειγμα, μία εφαρμογή ήχου χρησιμοποιεί μία υπηρεσία προσκηνίου για την αναπαραγωγή ενός ηχητικού κομματιού. Πρέπει να εμφανίζει ένα εικονίδιο στη γραμμή κατάστασης και συνεχίζει να εκτελείται ακόμα και όταν ο χρήστης δεν αλληλεπιδρά με την εφαρμογή.
- **Υπηρεσία Παρασκηνίου (Background Service)**: Εκτελεί μία λειτουργία που δεν είναι άμεσα εμφανής στο χρήστη. Για παράδειγμα, μία εφαρμογή που χρησιμοποιεί κάποια υπηρεσία για να συμπιέσει το χώρο της, συνήθως είναι υπηρεσία παρασκηνίου.

- **Δεμένη Υπηρεσία (Bound Service)**: Μία υπηρεσία είναι δεμένη όταν ένα εξάρτημα μίας εφαρμογής δένεται με αυτήν, κάνοντας κλήση της μεθόδου «bindService()». Προσφέρει μία διασύνδεση πελάτη-διακομιστή (client-server) που επιτρέπει στα εξαρτήματα να αλληλεπιδρούν με την υπηρεσία, να στέλνουν αιτήματα, να λαμβάνουν αποτελέσματα, με εφικτή τη διεπικοινωνία μεταξύ των διεργασιών (Inter-Process Communication - IPC). Τρέχει για όσο χρονικό διάστημα είναι συνδεδεμένη με ένα άλλο εξάρτημα της εφαρμογής. Πολλαπλά εξαρτήματα μπορούν να συνδεθούν με την ίδια υπηρεσία ταυτόχρονα, αλλά όταν όλα αυτά αποσυνδεθούν, τότε η υπηρεσία καταστρέφεται.

Σημείωση: Τα Services τρέχουν στο κύριο νήμα της διεργασίας που τα φιλοξενεί, χωρίς να δημιουργούν δικά τους νήματα.



Εικόνα 26 Ο κύκλος ζωής ενός Service. Το διάγραμμα στα αριστερά δείχνει τον κύκλο ζωής ενός Service που δημιουργείται με τη μέθοδο «`startService()`» και το διάγραμμα στα δεξιά δείχνει τον κύκλο ζωής ενός Service που δημιουργείται με τη μέθοδο «`bindService()`».

Στην παρούσα εφαρμογή γίνεται χρήση ενός Foreground Service για τη λήψη της τοποθεσίας του πιστοποιημένου χρήστη, με την ονομασία «`LocationService`», που κληρονομεί από την ασαφή (abstract) τάξη «`Service`».

Για να ξεκινήσει το Service καλούμε τη μέθοδο «`startService()`» μέσα από την τάξη «`BaseActivityFusedLocation`», ενώ για να το σταματήσουμε καλούμε τη μέθοδο «`stopService()`»:

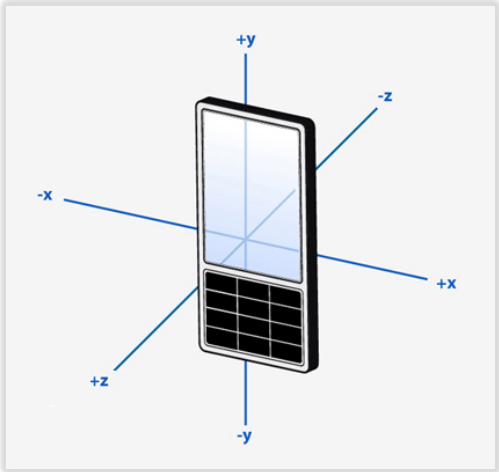
```
Intent intentService = new Intent(this, LocationService.class);
startService(intentService);
stopService(intentService);
```

Για να σταματήσει κάποιο Service από μόνο του και όχι μέσω κάποιου άλλου εξαρτήματος της εφαρμογής, τότε καλούμε τη μέθοδο «`stopSelf()`» μέσα από την τάξη του ίδιου του Service.

Για την ενημέρωση του περιβάλλοντος του χρήστη (user interface - UI), το LocationService αποστέλλει τοπικές αναμεταδόσεις (local broadcasts) στους δέκτες της εφαρμογής, διασφαλίζοντας την προστασία των δεδομένων (“Class LocalBroadcastManager”, 2017). Ως δέκτης έχει οριστεί η τάξη «`BaseActivityFusedLocation`», μέσω της οποίας η τάξη «`MainActivity`» αναλαμβάνει την ενημέρωση των fragments της Ταχύτητας, του Ταμπλό και του Χάρτη.

2.10 Επιταχυνσιόμετρο

Το **επιταχυνσιόμετρο (accelerometer)** είναι ένας ενσωματωμένος αισθητήρας σε κινητές συσκευές, ο οποίος έχει την ικανότητα να μετρά δυνάμεις επιτάχυνσης, μετατρέποντας τη φυσική ποσότητα σε ηλεκτρικό σήμα. Αυτές οι δυνάμεις μπορεί να είναι είτε στατικές, όπως η επιτάχυνση της βαρύτητας, είτε δυναμικές, όπως οι δυνάμεις που προκαλούνται από αλλαγές στην ταχύτητα ή στην διεύθυνση της κίνησης. Το επιταχυνσιόμετρο ανήκει στους **αισθητήρες κίνησης (motion sensors)** του Android και μετρά τις δυνάμεις που δέχεται η κινητή συσκευή σε κάθε δεδομένη χρονική στιγμή, επιτρέποντάς μας να αναγνωρίσουμε προς ποια κατεύθυνση μετακινεί ο χρήστης τη συσκευή. Η επιτάχυνση έχει φυσικές διαστάσεις μήκους προς χρόνο στο τετράγωνο και η τιμή της εκφράζεται ως ένα διάνυσμα τριών διαστάσεων, το οποίο αναπαριστά τις τιμές των επιταχύνσεων στους άξονες x, y και z:



Εικόνα 27 Διάνυσμα τριών διαστάσεων, αναπαριστώντας τις τιμές των επιταχύνσεων στους άξονες x, y, z.

Η μονάδα μέτρησης των τιμών που παίρνουμε από τον αισθητήρα επιτάχυνσης είναι στο Διεθνές Σύστημα Μονάδων (SI), δηλαδή το μέτρο ανά δευτερόλεπτο στο τετράγωνο (m/s^2).

Για να αποκτήσουμε πρόσβαση στους αισθητήρες της συσκευής, αρχικοποιούμε ένα στιγμιότυπο της τάξης «`SensorManager`». Ως σημείο αναφοράς, χρησιμοποιούμε τη μέθοδο

«getSystemService()», περνώντας ως όρισμα το όνομα της υπηρεσίας του συστήματος που θέλουμε να αποκτήσουμε πρόσβαση, δηλαδή τους αισθητήρες, «SENSOR_SERVICE»:

```
SensorManager sensorManager =
                (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

Στη συνέχεια, καλώντας τη μέθοδο «getDefaultSensor()» του SensorManager και περνώντας ως όρισμα τον τύπο του αισθητήρα που μας ενδιαφέρει, τον «TYPE_ACCELEROMETER», αποκτούμε πρόσβαση στο επιταχυνσιόμετρο της συσκευής:

```
Sensor accelerometer =
            sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Το interface «SensorEventListener», χρησιμοποιείται για τη λήψη ειδοποιήσεων από τον SensorManager, όταν υπάρχουν νέα δεδομένα αισθητήρων. Επομένως, κάνουμε εγγραφή (register) του αισθητήρα επιτάχυνσης στον SensorManager, για τη λήψη των ειδοποιήσεων αυτών, με συχνότητα δειγματοληψίας «SENSOR_DELAY_NORMAL», που αντιστοιχεί σε 200.000 microseconds:

```
sensorManager.registerListener(this, accelerometer,
                                    SensorManager.SENSOR_DELAY_NORMAL);
```

Στη συνέχεια, υλοποιώντας τη μέθοδο «onSensorChanged()» του interface «SensorEventListener», μέσω του αντικειμένου «sensorEvent», λαμβάνουμε τα νέα δεδομένα του αισθητήρα και τα αποθηκεύουμε στον στατικό πίνακα «sAcceleration[]», τύπου float (κινητής υποδιαστολής):

```
public static float sAcceleration[];

@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    // Get the type of this sensor.
    switch (sensorEvent.sensor.getType()){
        // An accelerometer sensor type (x, y, z).
        case Sensor.TYPE_ACCELEROMETER:
            // Get a copy of this object.
            sAcceleration = sensorEvent.values.clone();
            break;
        default:
            Log.e(LOG_TAG, "Unknown sensor type: " +
                                        sensorEvent.sensor.getType());
            break;
    }
}
```

Για τη διακοπή της λήψης των ειδοποιήσεων, καταργούμε την εγγραφή (unregister) του listener για όλους τους αισθητήρες, ως εξής:

```
sensorManager.unregisterListener(this);
```

2.10.1 Υπολογισμός Κραδασμών

Για τον υπολογισμό των κραδασμών που προκαλούνται κατά τη μετακίνηση του χρήστη, συγκρίνουμε το μέτρο της τωρινής επιτάχυνσης που λάβαμε από το επιταχυνσιόμετρο, με το μέτρο της προηγούμενης επιτάχυνσης και λαμβάνουμε την απόλυτη διαφορά τους, με μονάδα μέτρησης το m/s^2.

Το αποτέλεσμα που λαμβάνουμε είναι τύπου *double* (κινητής υποδιαστολής διπλής ακρίβειας), ίδιος τύπος με μία μεταβλητή τύπου *float*, αλλά με μεγαλύτερη ακρίβεια, περιέχοντας διπλό αριθμό δεκαδικών ψηφίων από τη μεταβλητή *float*. Εμάς όμως μας αρκούν μόλις τρία δεκαδικά ψηφία από τον αριθμό αυτό, επομένως, για να αποφύγουμε τη συλλογή περιττού όγκου δεδομένων, μετατρέπουμε τη μεταβλητή double σε αλφαριθμητικό (String), κρατώντας μόνο τα τρία δεκαδικά ψηφία.

Οι πράξεις αυτές υλοποιούνται μέσω της μεθόδου «`getShake()`», η οποία μας επιστρέφει το μέγεθος του κραδασμού ως αλφαριθμητικό:

```
public static float sAcceleration[] = {DEFAULT, DEFAULT, DEFAULT};
public static double sMagnitudeAccelerationLast = DEFAULT;
public static double sMagnitudeAccelerationCurrent = DEFAULT;

public static String getShake(){
  try{
    // Get the current magnitude of the device's total acceleration.
    sMagnitudeAccelerationCurrent = Math.sqrt(
                sAcceleration[0] * sAcceleration[0]
                + sAcceleration[1] * sAcceleration[1]
                + sAcceleration[2] * sAcceleration[2]);
    // If the last magnitude does not have the default value, then...
    if (sMagnitudeAccelerationLast != DEFAULT){
        // Get the difference between the last magnitude of the total
        // acceleration and the current.
        double delta = sMagnitudeAccelerationLast -
                                        sMagnitudeAccelerationCurrent;
        // Save the current magnitude.
        sMagnitudeAccelerationLast = sMagnitudeAccelerationCurrent;

        // Return the absolute value of the delta.
        return (new DecimalFormat("#.###", new
             DecimalFormatSymbols(Locale.UK)).format(Math.abs(delta)));
    }
    else{
        // Save the current magnitude.
        sMagnitudeAccelerationLast = sMagnitudeAccelerationCurrent;
        // Return the default value.
        return String.valueOf(DEFAULT);
    }
  }
  catch (Exception ex){
    Log.d(LOG_TAG, "Exception caught: " + ex);
    return String.valueOf(DEFAULT);
  }
}
```

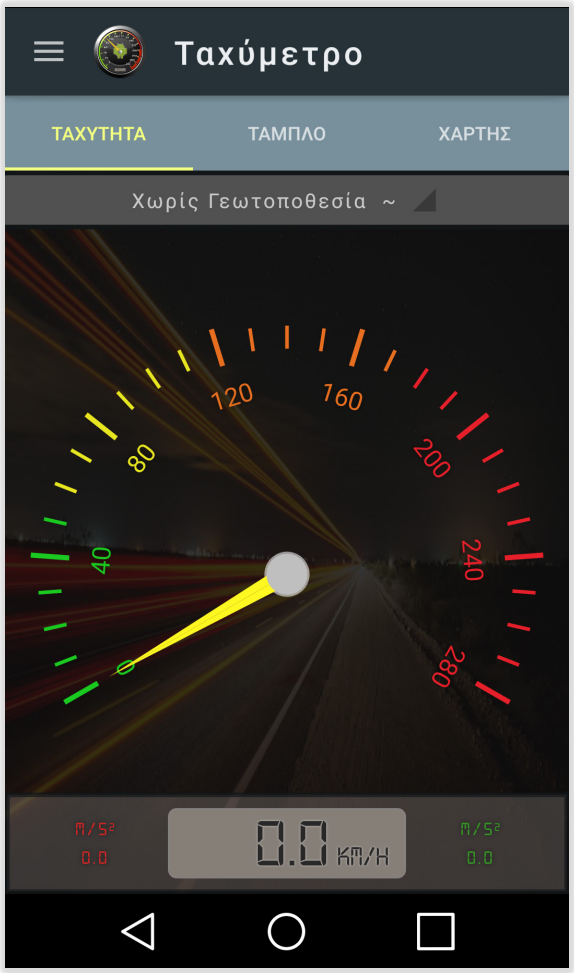
2.11 Μοτίβα Πλοήγησης

Η πλατφόρμα Android παρέχει διάφορα μοτίβα πλοήγησης που μπορούν να χρησιμοποιηθούν στις εφαρμογές, ούτως ώστε οι χρήστες να μπορούν να διασχίζουν αποτελεσματικά και ενστικτωδώς το περιεχόμενο της εκάστοτε εφαρμογές, απολαμβάνοντας υψηλή εμπειρία χρήσης (“Implementing Effective Navigation”, 2016).

- Τα μοτίβα αυτά, τα οποία εφαρμόζονται και στην παρούσα εφαρμογή, είναι:
  - Συρόμενες Προβολές με Καρτέλες (Swipe Views with Tabs).
  - Συρτάρι Πλοήγησης (Navigation Drawer).
  - Πάνω Πλοήγηση (Up Navigation).
  - Πίσω Πλοήγηση (Back Navigation).

2.11.1 Συρόμενες Προβολές με Καρτέλες

Οι **Συρόμενες Προβολές (Swipe Views)** παρέχουν πλευρική πλοήγηση μεταξύ αδελφών οθονών. Μία τέτοια πλοήγηση μπορεί να επιτευχθεί μέσω των **Καρτελών (Tabs)** με χρήση της οριζόντιας χειρονομίας δακτύλων (horizontal finger gesture).



Εικόνα 28 Συρόμενες Προβολές με Καρτέλες.

Η σχεδίαση των προβολών με δυνατότητα ολίσθησης γίνεται μέσω του στοιχείου «ViewPager», ενώ των καρτελών μέσω του γραφικού στοιχείου «TabLayout» (διαθέσιμα και τα δύο μέσω των Βιβλιοθηκών Υποστήριξης):

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:id="@+id/sliding_tabs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:tabGravity="fill"
        app:tabMaxWidth="0dp"
        app:tabMode="fixed" />

</android.support.v4.view.ViewPager>
```

Για να εισάγουμε προβολές παιδιών που αντιπροσωπεύουν κάθε σελίδα στο «ViewPager», θα πρέπει να συνδέσουμε τη διάταξη αυτή σε έναν «PagerAdapter». Οι δύο διαθέσιμοι προσαρμογείς είναι:

- FragmentPagerAdapter: Προσαρμογέας ιδανικός για την πλοήγηση ανάμεσα σε οθόνες αδελφών, που αντιπροσωπεύουν ένα μικρό και σταθερό αριθμό σελίδων.
- FragmentStatePagerAdapter: Προσαρμογέας ιδανικός για την αναζήτηση σε μία συλλογή αντικειμένων, με απροσδιόριστο αριθμό σελίδων. Καθώς ο χρήστης περιηγείται σε άλλες σελίδες, ο προσαρμογέας καταστρέφει fragments που δεν είναι ορατά, ελαχιστοποιώντας τη χρήση της μνήμης.

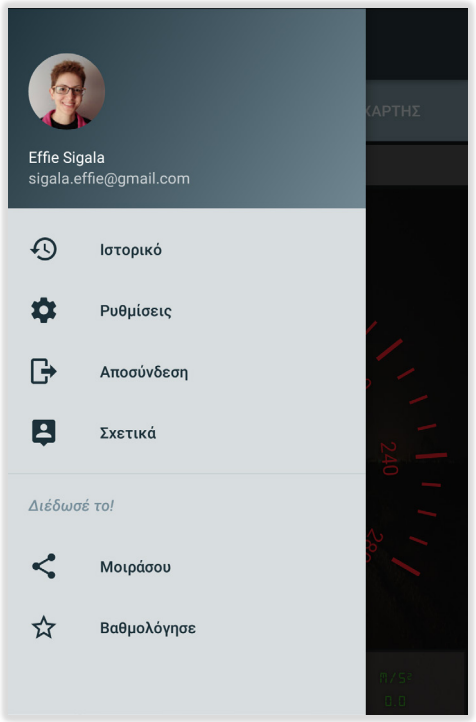
Στην παρούσα εφαρμογή γίνεται χρήση του «FragmentPagerAdapter», καθώς οι οθόνες που πρόκειται να εισαχθούν είναι τρεις (Ταχύτητα, Ταμπλό, Χάρτης). Η υλοποίηση του προσαρμογέα γίνεται μέσω της custom τάξης «SectionsPagerAdapter», που κληρονομεί από την αφηρημένη τάξη «FragmentPagerAdapter», η οποία με τη σειρά της κληρονομεί από την τάξη «PagerAdapter» (παιδί της υπερτάξης «Object»).

Η σύνδεση του «ViewPager» με τα «Tabs» γίνεται μέσω της μεθόδου «setupWithViewPager()», όπου η διάταξη των «Tabs» συμπληρώνεται αυτόματα από τους τίτλους της σελίδας του «PagerAdapter»:

```
// Retrieve the ViewPager from the UI.
ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
// Attach the ViewPager to the adapter.
viewPager.setAdapter(new SectionsPagerAdapter(
                        getSupportFragmentManager(), MainActivity.this));
// Retrieve the TabLayout from the UI.
TabLayout tabLayout = (TabLayout) findViewById(R.id.sliding_tabs);
// Connect the pager with the tabs.
tabLayout.setupWithViewPager(viewPager);
```

2.11.2 Συρτάρι Πλοήγησης

Το **Συρτάρι Πλοήγησης (Navigation Drawer)** είναι ένα πάνελ στο αριστερό άκρο της οθόνης, με τις κύριες επιλογές πλοήγησης της εφαρμογής.



Εικόνα 29 Συρτάρι Πλοήγησης στο αριστερό άκρο της οθόνης.

Όταν το Συρτάρι Πλοήγησης είναι κρυμμένο, τότε στο αριστερό άκρο της μπάρας της εφαρμογής εμφανίζεται ένα εικονίδιο Hamburger. Πατώντας πάνω στο εικονίδιο αυτό, το Συρτάρι Πλοήγησης αποκαλύπτεται.

Εικόνα 30 Εικονίδιο Hamburger στην μπάρα της εφαρμογής.

Η σχεδίαση του Συρταριού Πλοήγησης γίνεται μέσω του γραφικού στοιχείου «`NavigationView`», το οποίο ως παιδί (child view), εμπερικλείεται στη ριζική προβολή του γραφικού στοιχείου «`DrawerLayout`» (διαθέσιμα και τα δύο μέσω των Βιβλιοθηκών Υποστήριξης):

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@drawable/background_speed_scaled"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="@color/backgroundColorNavDrawer"
        android:fitsSystemWindows="true"
        android:theme="@style/SigalaTheme.NavigationView"
        app:headerLayout="@layout/nav_header_main"
        app:itemIconTint="@color/side_nav_bar_end_color"
        app:itemTextColor="@color/side_nav_bar_end_color"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Στην παραπάνω διάταξη, συμπεριλαμβάνεται ένα ακόμα παιδί (child view), το οποίο αναφέρεται στο αρχείο πόρων, τύπου «layout», με το όνομα «`app_bar_main.xml`».

Τα περιεχόμενα του μενού λαμβάνονται από το αρχείο πόρων, τύπου «`menu`», με το όνομα «`activity_main_drawer.xml`»:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_log_in"
            android:icon="@drawable/ic_login_24dp"
            android:title="@string/nav_log_in" />
        <item
            android:id="@+id/nav_history"
            android:icon="@drawable/ic_history_24dp"
            android:title="@string/nav_history" />
        <item
            android:id="@+id/nav_settings"
            android:icon="@drawable/ic_settings_24dp"
            android:title="@string/nav_settings" />
        <item
            android:id="@+id/nav_log_out"
            android:icon="@drawable/ic_logout_24dp"
            android:title="@string/nav_log_out" />
    </group>

    <group android:checkableBehavior="single" >
        <item
```

```
            android:id="@+id/nav_about"
            android:icon="@drawable/ic_about_24dp"
            android:title="@string/nav_about" />
    </group>

    <item android:title="@string/nav_spread_the_word"
        android:id="@+id/nav_spread_the_word">
        <menu>
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_share_24dp"
                android:title="@string/nav_share" />
            <item
                android:id="@+id/nav_rate"
                android:icon="@drawable/ic_rate_24dp"
                android:title="@string/nav_rate" />
        </menu>
    </item>

</menu>
```

2.11.3 Πάνω Πλοήγηση

Η **Πάνω Πλοήγηση (Up Navigation)** δίνει τη δυνατότητα στο χρήστη να πλοηγηθεί από τις διάφορες οθόνες της εφαρμογής, πίσω στην αρχική οθόνη, δηλαδή στη γονική οθόνη στην ιεραρχία της εφαρμογής, με το πάτημα του κουμπιού “Πάνω”. Το κουμπί αυτό βρίσκεται στο αριστερό τμήμα της μπάρας ενεργειών (action bar), σε κάθε οθόνη που δεν είναι η γονική.

Εικόνα 31 Το κουμπί “Πάνω” στην μπάρα ενεργειών.

Για την υλοποίηση της “Πάνω” πλοήγησης, αρχικά δηλώνουμε στο **manifest** τις σχέσεις γονέα-παιδιού, στα μεταδεδομένα κάθε παιδιού. Στην παρούσα εφαρμογή η ιεραρχία των δραστηριοτήτων έχει ως εξής:

```
MainActivity
|-- AboutActivity
|-- HistoryActivity
|-- SettingsActivity
```

Παρακάτω φαίνονται οι σχέσεις γονέα-παιδιού για τα τρία παιδιά-Activities, όπως αυτές έχουν δηλωθεί στο manifest:

```
<!-- Activity: "About" -->
<activity
    android:name=".ui.AboutActivity"
    android:label="@string/activity_about">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
```

```
        android:value=".ui.MainActivity" />
</activity>

<!-- Activity: "History" -->
<activity android:name=".ui.HistoryActivity"
    android:label="@string/nav_history">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".ui.MainActivity" />
</activity>

<!-- Activity: "Settings" -->
<activity
    android:name=".ui.SettingsActivity"
    android:label="@string/settings_title"
    android:theme="@style/PreferencesTheme">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".ui.MainActivity" />
</activity>
```

Για να εμφανίσουμε το κουμπί “Πάνω” στην μπάρα ενεργειών προγραμματιστικά, ορίζουμε την τιμή «`true`» στη μέθοδο «`setDisplayHomeAsUpEnabled()`»:

```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

Τέλος, για να μπορέσει ο χρήστης να πλοηγηθεί από την τρέχουσα οθόνη στην προηγούμενη, πατώντας το κουμπί “Πάνω”, χρησιμοποιούμε τη μέθοδο «`navigateUpFromSameTask()`», όπου το ID για τη δράση αυτή, από προεπιλογή, είναι το «`android.R.id.home`»:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // Respond to the action bar's Up/Home button...
        case android.R.id.home:
            NavUtils.navigateUpFromSameTask(this);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Η λειτουργία εκκίνησης (launch mode) της γονικής δραστηριότητας «`MainActivity`» έχει λάβει την τιμή <`singleTop`> στο manifest, συνεπώς, μόλις κληθεί η παραπάνω μέθοδος, η γονική δραστηριότητα «`MainActivity`» θα μεταφερθεί στην κορυφή της στοίβας. Σε αντίθετη περίπτωση, θα αποσπόταν από τη στοίβα και θα δημιουργούταν ένα νέο στιγμιότυπο αυτής, στην κορυφή της στοίβας.

2.11.4 Πίσω Πλοήγηση

Η **Πίσω Πλοήγηση (Back Navigation)** είναι ο τρόπος με τον οποίο οι χρήστες μετακινούνται προς τα πίσω, μέσω του ιστορικού των οθονών που επισκέφτηκαν στο παρελθόν. Το κουμπί της επιστροφής δεν το προσθέτουμε στο περιβάλλον του χρήστη, καθώς όλες οι συσκευές Android διαθέτουν ένα κουμπί για το συγκεκριμένο τύπο πλοήγησης.

Εικόνα 32 Το κουμπί "Πίσω" στην μπάρα πλοήγησης (το τριγωνικό σύμβολο στα αριστερά).

Το σύστημα διατηρεί μία πίσω στοίβα (back stack) από Activities καθώς ο χρήστης περιηγείται στην εφαρμογή, επιτρέποντας στο σύστημα να πλοηγηθεί προς τα πίσω σωστά με το πάτημα του κουμπιού "Πίσω". Παρ' όλα αυτά, υπάρχουν περιπτώσεις που η εφαρμογή πρέπει να καθορίσει με μη αυτόματο τρόπο τη συμπεριφορά του κουμπιού αυτού, ούτως ώστε να προσφέρει την καλύτερη εμπειρία χρήστη.

Στην παρούσα εφαρμογή, χρειάζεται να παρακάμψουμε την προεπιλεγμένη λειτουργία του κουμπιού "Πίσω" όταν ο χρήστης βρίσκεται στην αρχική οθόνη, προσθέτοντας τις παρακάτω λειτουργίες:

- Όταν το Συρτάρι Πλοήγησης είναι ανοιχτό, τότε με το πάτημα του κουμπιού "Πίσω" το Συρτάρι Πλοήγησης κλείνει.
- Όταν το Συρτάρι Πλοήγησης είναι κλειστό, τότε με το πάτημα του κουμπιού "Πίσω":
  - Εάν η εφαρμογή δεν έχει οριστεί να λειτουργεί στο παρασκήνιο, τότε η εφαρμογή καταστρέφεται.
  - Εάν η εφαρμογή έχει οριστεί να λειτουργεί στο παρασκήνιο, τότε η εφαρμογή μεταφέρεται στο πίσω μέρος της στοίβας με τα Activities.

```
@Override
public void onBackPressed() {
    Log.d(LOG_TAG, "onBackPressed()");
    // If the Navigation Drawer is open, then close it...
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
    }
    // If the Navigation Drawer is not open, then...
    else {

        // If the Speedometer is not set to work in the background...
        if (!mSharedPref.getBoolean(
                    getString(R.string.settings_work_in_background_key),
                                              DEFAULT_WORK_IN_BACKGROUND))
        {
            // Finish the current activity.
            super.onBackPressed();
        }
        // The Speedometer is set to work in the background...
        else {
            // Move the task containing this activity to the
```

```
            // back of the activity stack.
            moveTaskToBack(true);
        }
    }
}
```

2.12 Γρήγορη Προσπέλαση Δεδομένων

Το Android δίνει τη δυνατότητα στους προγραμματιστές να αποθηκεύουν πρωταρχικούς τύπους δεδομένων σε διατηρούμενα (persistent) ζεύγη **κλειδιού-τιμής (key-value)**. Η τεχνική αυτή επιτυγχάνεται μέσω του **SharedPreferences** API. Ενδείκνυται για την αποθήκευση μικρού όγκου δεδομένων και για δεδομένα που απαιτείται γρήγορη προσπέλαση (π.χ. προτιμήσεις του χρήστη που αφορούν την εφαρμογή).

Τα ζεύγη διατηρούνται έως ότου:

- τα σβήσουμε προγραμματιστικά (π.χ. κατά την αποσύνδεση του χρήστη από την εφαρμογή) ή
- κάνοντας εκκαθάριση των δεδομένων της εφαρμογής μέσα από τις ρυθμίσεις της συσκευής ή
- απεγκαθιστώντας την εφαρμογή.

Το *κλειδί* είναι τύπου String και αντιπροσωπεύει το όνομα της προτίμησης που θέλουμε να τροποποιήσουμε ή να ανακτήσουμε.

Η *τιμή* μπορεί να είναι τύπου boolean, float, int, long, String ή Set<String>.

Στην παρούσα εφαρμογή μία από τις αλφαριθμητικές τιμές (String) που αποθηκεύουμε στα Shared Preferences είναι το ID του πιστοποιημένου χρήστη, ούτως ώστε να παραμένει συνδεδεμένος στην εφαρμογή, χωρίς να χρειάζεται κάποια επιπρόσθετη ενέργεια από τον ίδιο κάθε φορά που εκτελεί την εφαρμογή.

Για να αποκτήσουμε πρόσβαση στα δεδομένα προτιμήσεων, αρχικοποιούμε ένα στιγμιότυπο της τάξης SharedPreferences. Το στιγμιότυπο αυτό δείχνει στο προεπιλεγμένο αρχείο που χρησιμοποιείται από το framework προτιμήσεων της εφαρμογής αυτής.

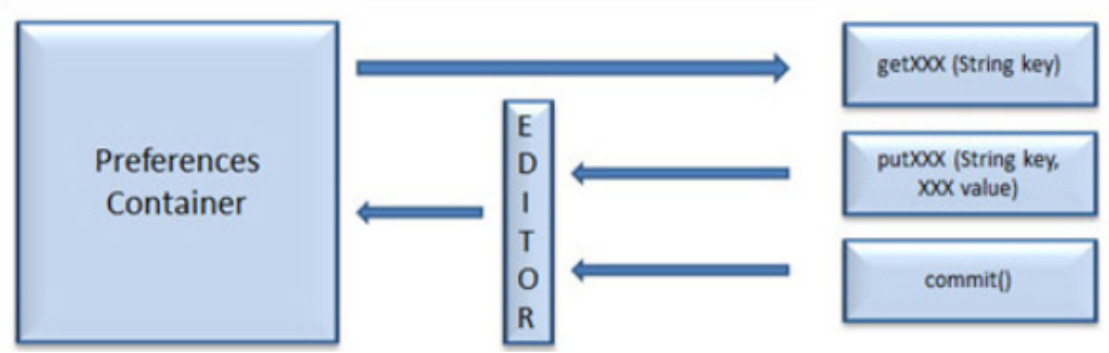
```
SharedPreferences mSharedPref = PreferenceManager
                  .getDefaultSharedPreferences(getApplicationContext());
```

Για να καταχωρήσουμε ένα νέο ζεύγος κλειδί-τιμή ή να τροποποιήσουμε ένα υπάρχον στα δεδομένα προτιμήσεων, δημιουργούμε έναν Editor, λαμβάνοντας ένα νέο στιγμιότυπο του interface «`SharedPreferences.Editor`». Μέσω της μεθόδου «`putString(String key, String value)`» δηλώνουμε μία αλφαριθμητική τιμή στον Editor των προτιμήσεων και με την εντολή «`commit()`» τις αλλαγές που κάναμε στον Editor τις οριστικοποιούμε στο στιγμιότυπο SharedPreferences που επεξεργαζόμαστε, σε αντίθετη περίπτωση οι αλλαγές χάνονται.

```
SharedPreferences.Editor editor = mSharedPref.edit();
editor.putString(MyConstants.SHARED_GOOGLE_UID, uid);
editor.commit();
```

Για να ανακτήσουμε το ID του χρήστη από τα δεδομένα προτιμήσεων, δηλαδή την αλφαριθμητική τιμή κάποιου ζεύγους, χρησιμοποιούμε τη μέθοδο «`getString(String key, String devValue)`», δίνοντας ως πρώτο όρισμα το κλειδί του ζεύγους που αναζητούμε και ως δεύτερο όρισμα μία προεπιλεγμένη τιμή, σε περίπτωση που το ζεύγος αυτό δεν υπάρχει.

```
String uID = mSharedPref.getString(MyConstants.SHARED_GOOGLE_UID, "");
```



Εικόνα 33 SharedPreferences API: Ανάκτηση και καταχώρηση δεδομένων.

Αντίστοιχες μεθόδους με τις `putString()` / `getString()` υπάρχουν και για τους υπόλοιπους τύπους δεδομένων που υποστηρίζουν τα Shared Preferences, με τις αντίστοιχες ονομασίες.

2.13 Ρυθμίσεις Εφαρμογής

Οι περισσότερες εφαρμογές περιλαμβάνουν ρυθμίσεις, που επιτρέπουν στους χρήστες να τροποποιούν τις λειτουργίες και τις συμπεριφορές τους.

Στην παρούσα εφαρμογή, οι χρήστες έχουν τη δυνατότητα να τροποποιήσουν τις ακόλουθες ρυθμίσεις:

- Εργασία στο παρασκήνιο (λήψη τοποθεσίας όταν η εφαρμογή δεν είναι σε πρώτο πλάνο).
- Κατάσταση λειτουργίας (η συχνότητα λήψης των αλλαγών της τοποθεσίας του χρήστη, η οποία επηρεάζει άμεσα την κατανάλωση μπαταρίας).
- Αυτόματη περιστροφή του χάρτη σύμφωνα με την κατεύθυνση του χρήστη.

Για τη δημιουργία ενός περιβάλλοντος, που παρέχει συνέπεια στην εμπειρία χρήστη με άλλες εφαρμογές Android, καθώς και με τις ρυθμίσεις του συστήματος, προτείνεται η χρήση των Preference APIs της πλατφόρμας Android (“Settings”, 2017). Το περιβάλλον χρήστη δε δημιουργείται με χρήση των αντικειμένων View, όπως στα Activities, αλλά χρησιμοποιώντας διάφορες υποτάξεις της τάξης «`Preference`», οι οποίες δηλώνονται σε ένα αρχείο XML.

Ένα αντικείμενο Preference είναι το δομικό στοιχείο για μία ξεχωριστή ρύθμιση, το οποίο εμφανίζεται ως στοιχείο μίας λίστας, παρέχοντας το κατάλληλο περιβάλλον χρήστη για την τροποποίηση της ρύθμισης αυτής.

Κάθε Preference που προστίθεται στο αρχείο XML έχει ένα αντίστοιχο ζευγάρι κλειδιού-τιμής, το οποίο αποθηκεύεται συστημικά στο προεπιλεγμένο αρχείο «`SharedPreferences`» των ρυθμίσεων της εφαρμογής. Κάθε αλλαγή που κάνει ο χρήστης στις ρυθμίσεις, το σύστημα

ενημερώνει αυτόματα την αντίστοιχη τιμή στο αρχείο «SharedPreferences». Η μόνη στιγμή που αλληλεπιδρούμε άμεσα με το αρχείο αυτό, είναι για την ανάγνωση κάποιας τιμής, με σκοπό να προσδιορίσουμε τη συμπεριφορά της εφαρμογής με βάση τη ρύθμιση του χρήστη.

Για την εμφάνιση της λίστας, θα πρέπει να δημιουργήσουμε ένα «Activity», το οποίο κληρονομεί από την τάξη «AppCompatActivity» και φιλοξενεί ένα «PreferenceFragment». Το «PreferenceFragment» είναι υποτάξη (subclass) της αφηρημένης τάξης «Fragment» και είναι υπεύθυνο για την εμφάνιση των ρυθμίσεων της εφαρμογής.

Σημείωση: Εάν η εφαρμογή υποστηρίζει εκδόσεις του Android προγενέστερες της έκδοσης 3.0 (δηλαδή από το επίπεδο API 10 και πίσω), τότε το «Activity» θα πρέπει να κληρονομεί από την τάξη «PreferenceActivity».

Στην παρούσα εφαρμογή χρησιμοποιούνται τα κάτωθι αντικείμενα Preference:

- SwitchPreference: Παρέχει τη δυνατότητα εναλλαγής δύο καταστάσεων ως γραφικό στοιχείο διακόπτη (switch).
- ListPreference: Εμφανίζει μία λίστα καταχωρήσεων ως παράθυρο διαλόγου.
- CheckBoxPreference: Παρέχει τη δυνατότητα εναλλαγής δύο καταστάσεων ως γραφικό στοιχείο πλαισίου ελέγχου (checkbox).

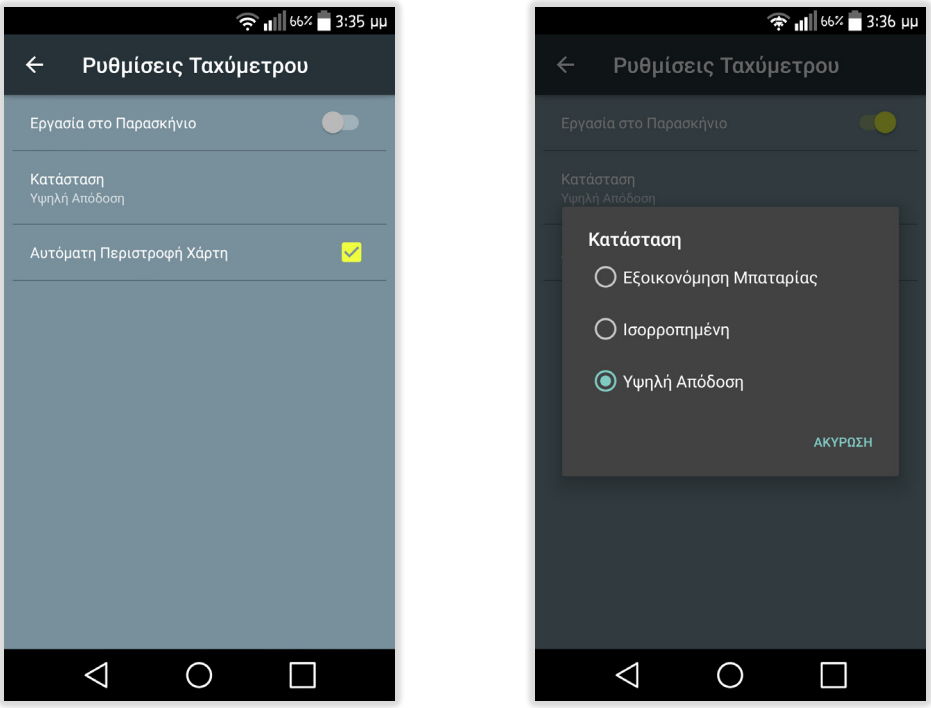
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:title="@string/settings_title">

    <!-- Preferences: Work in Background -->
    <SwitchPreference
        android:defaultValue="false"
        android:key="@string/settings_work_in_background_key"
        android:title="@string/settings_work_in_background_label" />

    <!-- Preferences: Mode -->
    <ListPreference
        android:defaultValue="@string/settings_mode_default"
        android:entries="@array/settings_mode_labels"
        android:entryValues="@array/settings_mode_values"
        android:key="@string/settings_mode_key"
        android:title="@string/settings_mode_label" />

    <!-- Preferences: Auto Rotate Map -->
    <CheckBoxPreference
        android:defaultValue="true"
        android:key="@string/settings_auto_rotate_map_key"
        android:title="@string/settings_auto_rotate_map_label" />

</PreferenceScreen>
```



Εικόνα 34 Προεπιλεγμένες ρυθμίσεις (αριστερά). Παράθυρο διαλόγου με τις καταστάσεις λειτουργίας (δεξιά).

Εικόνα 35 Ενεργοποιημένος διακόπτης (αριστερά). Απενεργοποιημένος διακόπτης (δεξιά).

Εικόνα 36 Επιλεγμένο πλαίσιο ελέγχου (αριστερά). Μη επιλεγμένο πλαίσιο ελέγχου (δεξιά).

2.14 Ασύγχρονη Διαχείριση Δεδομένων

Για την προβολή του ιστορικού των διαδρομών του χρήστη, θα πρέπει να γίνει ανάκτηση ενός μεγάλου όγκου πληροφοριών από τη βάση δεδομένων, που αφορούν το συγκεκριμένο χρήστη, για μία συγκεκριμένη ημέρα. Εν συνεχεία, τα δεδομένα αυτά θα πρέπει να διαχειριστούν κατάλληλα, ούτως ώστε να απεικονιστεί η διαδρομή πάνω στο χάρτη, η κάμερα να εστιάσει στο μέγιστο δυνατό, να τοποθετηθούν τα σημεία αφετηρίας και τερματισμού και να μπορούν να απεικονιστούν οι κραδασμοί.

Εάν η διαχείριση των δεδομένων γίνει στο UI thread, δηλαδή στο κύριο νήμα, το οποίο είναι υπεύθυνο για το γραφικό περιβάλλον του χρήστη και για την εξυπηρέτηση των αλληλεπιδράσεων του χρήστη με την εφαρμογή, τότε ενδέχεται να παγώσει η εφαρμογή κατά τη διεργασία του μεγάλου όγκου δεδομένων, προκαλώντας την κατάρρευσή της (crash). Για το λόγο αυτό γίνεται χρήση της **Ασύγχρονης Διεργασίας** (“AsyncTask”, 2017), η οποία με αρκετά απλό και εύκολο τρόπο, επιτρέπει την εκτέλεση εργασιών στο παρασκήνιο και την εμφάνιση των αποτελεσμάτων στο UI thread, όταν αυτό κριθεί απαραίτητο, χωρίς να χρειάζεται να διαχειριστούμε threads ή handlers.

- Μία ασύγχρονη διεργασία μπορεί να τροφοδοτηθεί με τρεις τύπους ορισμάτων:
  1) **Παράμετροι (Params)**: Ο τύπος των παραμέτρων που αποστέλλονται στην διεργασία κατά την εκτέλεσή της.
  2) **Πρόοδος (Progress)**: Ο τύπος των μονάδων προόδου που εμφανίζονται στο UI thread, κατά τη διάρκεια υπολογισμού, δηλαδή των εντολών που εκτελούνται στο παρασκήνιο.
  3) **Αποτέλεσμα (Result)**: Ο τύπος του αποτελέσματος του υπολογισμού που έγινε στο παρασκήνιο.

  Εάν ένας τύπος παραληφθεί, τότε χρησιμοποιούμε τον τύπο Void.

- Η ασύγχρονη διεργασία περνάει μέσα από τέσσερα βήματα:
  1) `onPreExecute()`: Καλείται στο UI thread πριν εκτελεστεί η διεργασία. Αυτό το βήμα χρησιμοποιείται συνήθως για τη ρύθμιση της διεργασίας, για παράδειγμα, εμφανίζοντας μία γραμμή προόδου στο user interface.
  2) `doInBackground(Params...)`: Καλείται στο νήμα που εκτελείται στο παρασκήνιο, αμέσως μετά την εκτέλεση του `onPreExecute()`. Αυτό το βήμα χρησιμοποιείται για την εκτέλεση των εντολών στο παρασκήνιο, που μπορεί να διαρκέσει αρκετή ώρα. Οι παράμετροι της ασύγχρονης διεργασίας μεταβιβάζονται σε αυτό το βήμα. Το αποτέλεσμα του υπολογισμού επιστρέφεται μέσω αυτού του βήματος και μεταφέρεται στο τελευταίο. Το βήμα αυτό μπορεί επίσης να καλέσει τη μέθοδο `publishProgress(Progress...)` για να εμφανίσει μία ή περισσότερες μονάδες προόδου στο UI thread, μέσω του βήματος `onProgressUpdate(Progress...)`.
  3) `onProgressUpdate(Progress...)`: Καλείται στο UI thread, μετά από μία κλήση της μεθόδου `publishProgress(Progress...)`. Ο χρόνος εκτέλεσής της δεν είναι καθορισμένος. Η μέθοδος αυτή χρησιμοποιείται για την εμφάνιση οποιασδήποτε μορφής προόδου στο user interface, ενώ ο υπολογισμός στο παρασκήνιο εξακολουθεί να εκτελείται. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για την κίνηση μιας γραμμής προόδου.
  4) `onPostExecute(Result)`: Καλείται στο UI thread μετά την ολοκλήρωση του υπολογισμού που εκτελέστηκε στο παρασκήνιο και το αποτέλεσμα του παρασκηνιακού υπολογισμού μεταβιβάζεται στο βήμα αυτό ως παράμετρος.

Για τις ανάγκες της παρούσας εφαρμογής, χρησιμοποιούμε μόνο το 2ο και το 4ο βήμα. Η παράμετρος με την οποία τροφοδοτείτε η μέθοδος «`doInBackground()`», είναι ένα στιγμιότυπο τύπου «`DataSnapshot`», που αντιπροσωπεύει τα δεδομένα που ανακτήθηκαν από τη βάση δεδομένων για το συγκεκριμένο χρήστη, για μία συγκεκριμένη ημέρα. Στο παρασκήνιο εκτελείται ένα πλήθος εντολών που αφορούν τη δημιουργία της διαδρομής που ακολούθησε ο χρήστης τη συγκεκριμένη ημέρα, τοποθετώντας τα σημεία αφετηρίας και τερματισμού και ομαδοποιώντας τους κραδασμούς σε τρεις λίστες, ανάλογα με την έντασή τους. Ως αποτέλεσμα, επιστρέφει ένα αντικείμενο τύπου «`CameraUpdate`», το οποίο τροφοδοτεί τη μέθοδο «`onPostExecute()`» και είναι υπεύθυνο για τον τρόπο με τον οποίο θα κινηθεί η κάμερα, εστιάζοντας το μέγιστο δυνατό στη διαδρομή που έχει απεικονιστεί πάνω στο χάρτη. Η απεικόνιση της διαδρομής στο χάρτη γίνεται μέσω της τάξης «`PolylineOptions`», ενώ των κραδασμών μέσω της τάξης «`HeatmapTileProvider`»:

```
private class HistoryAsyncTask extends AsyncTask<DataSnapshot, Void,
                                                  CameraUpdate> {


    private PolylineOptions mPolylineOptions;
    private HeatmapTileProvider mHeatmapTileProviderGood;
    private HeatmapTileProvider mHeatmapTileProviderMedium;
```

```
    private HeatmapTileProvider mHeatmapTileProviderBad;

    @Override
    protected CameraUpdate doInBackground(DataSnapshot...
                                                    dataSnapshots) {
        // Initialize a Polyline with pairs of Latitude and Longitude
        // coordinates and define the camera move with bounds and
        // padding.
        try{

            // Use a PolylineOptions object to add points to it.
            mPolylineOptions = new PolylineOptions();

            // Create a builder that is able to create a minimum bound
            // based on a set of LatLng points.
            LatLngBounds.Builder latLngBoundsBuilder = new
                                            LatLngBounds.Builder();

            // The "dataSnapshots[0]" is the node of the requested
            // date. Every child of the node "dataSnapshots[0]" is a
            // node representing a "timestamp".
            for (DataSnapshot nodeTimestamp :
                                    dataSnapshots[0].getChildren()) {

                // Represent a pair of latitude and longitude
                // coordinates, stored as degrees.
                LatLng latLng;

                // Every child of the node "nodeTimestamp" consists of
                // information regarding this timestamp.
                for (DataSnapshot childContent :
                                        nodeTimestamp.getChildren()) {
                    // Get the LATITUDE.
                    // Get the LONGITUDE.
                    // Get the SHAKE.
                }

                // Set the "Start" marker.
                // Set the "Finish" marker.

                // Add this vertex to the end of the polyline being
                // built.
                mPolylineOptions.add(latLng);

                // Include this point for building of the bounds.
                latLngBoundsBuilder.include(latLng);
            }

            // Initialize the shakes.
            initHeatMapTileShake();

            // Create the LatLng bounds to set into the map camera.
            LatLngBounds latLngBounds = latLngBoundsBuilder.build();

            // Define the camera move with bounds and padding to set
```

```
            // into map.
            CameraUpdate cameraUpdate = CameraUpdateFactory
                    .newLatLngBounds(latLngBounds, ZOOM_CAMERA_PADDING);

            return cameraUpdate;
        }
        catch (Exception ex){
            Log.e(LOG_TAG, " Exception caught: " + ex);
            return null;
        }
    }

    /**
     * Update the screen with the given CameraUpdate (which
     * was the result of the HistoryActivity.HistoryAsyncTask).
     */
    @Override
    protected void onPostExecute(CameraUpdate cameraUpdate) {
        // Hide the loading indicator.
        hideLoadingIndicator();
        try{
            // Add the Markers and the Polyline to the map.
            addMarkersAndPolyline();
            // Add the endpoints on the map.
            addEndPoints();
            // Add the shakes on the map.
            addShakes();
            // Animate the movement of the camera from the current
            // position to the position defined.
            mGoogleMap.animateCamera(cameraUpdate);
            // Hide the TextView located at the top of the map.
            tvNoData.setVisibility(View.INVISIBLE);
        }
        catch (Exception ex){
            Log.e(LOG_TAG, "Exception caught: " + ex);
        }
    }
}
```

Η εκτέλεση της ασύγχρονης εργασίας γίνεται μέσω της εντολής «`execute()`», περνώντας ως όρισμα τα δεδομένα που ανακτήθηκαν από τη βάση δεδομένων για το συγκεκριμένο χρήστη, για μία συγκεκριμένη ημέρα:

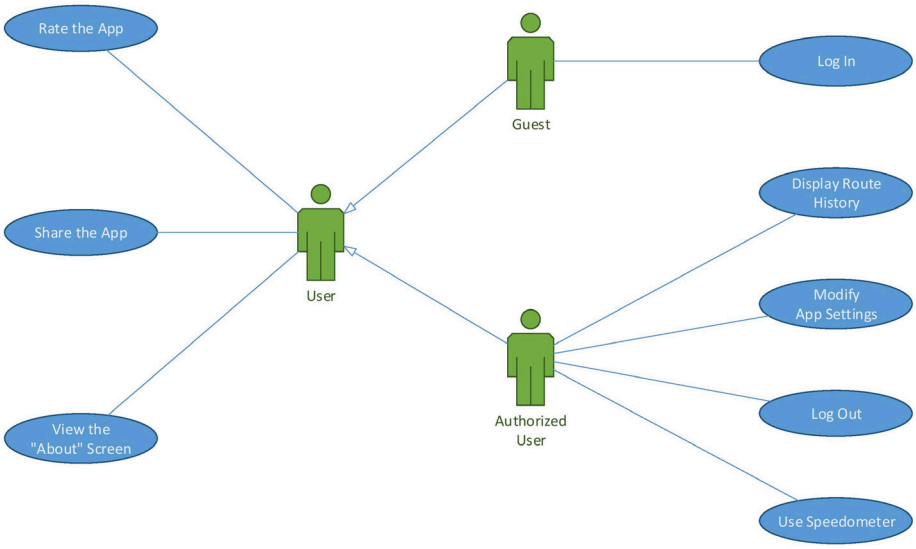
```
new HistoryAsyncTask().execute(dataSnapshot);
```

2.15 Διαγράμματα Περιπτώσεων Χρήσης

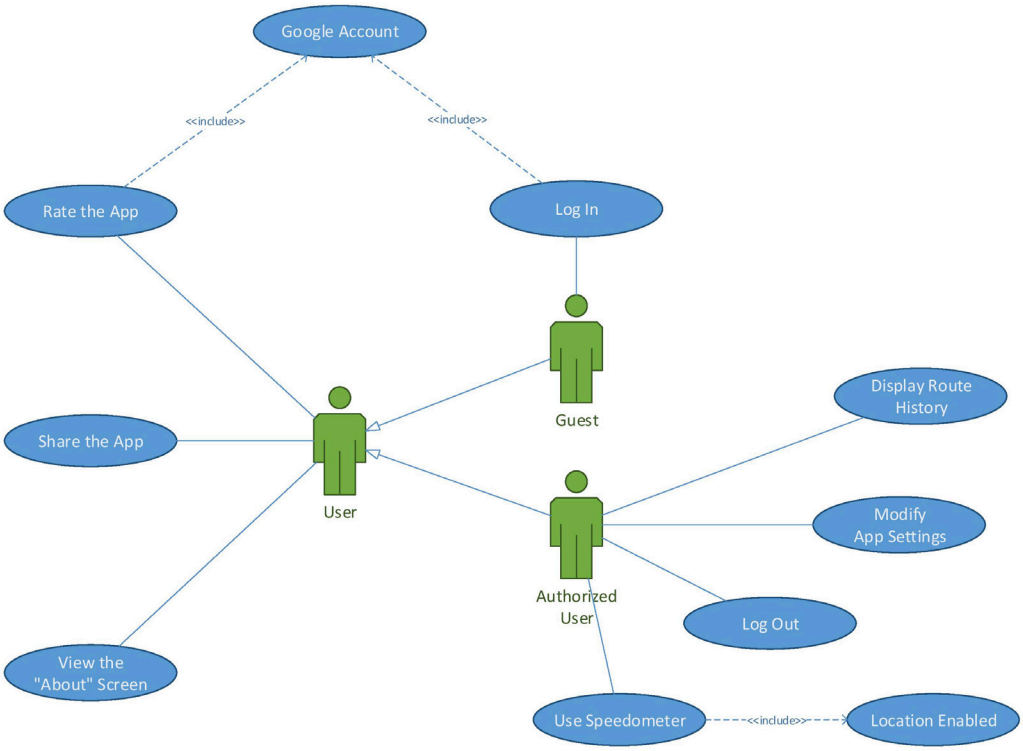
Για την ανάλυση των απαιτήσεων και το σχεδιασμό του λογισμικού μπορούν να χρησιμοποιηθούν τα **Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)**, τα οποία αναπαριστούν τις λειτουργίες ενός συστήματος από την οπτική γωνία του χρήστη.

Στη συνέχεια παρουσιάζονται οι τρεις πρώτες φάσεις δόμησης του έργου σε σχέση με το χρόνο (Gornik, 2011), ακολουθώντας το αντικειμενοστρεφές μοντέλο **Rational Unified Process - RUP** (Kruchten, 2003):

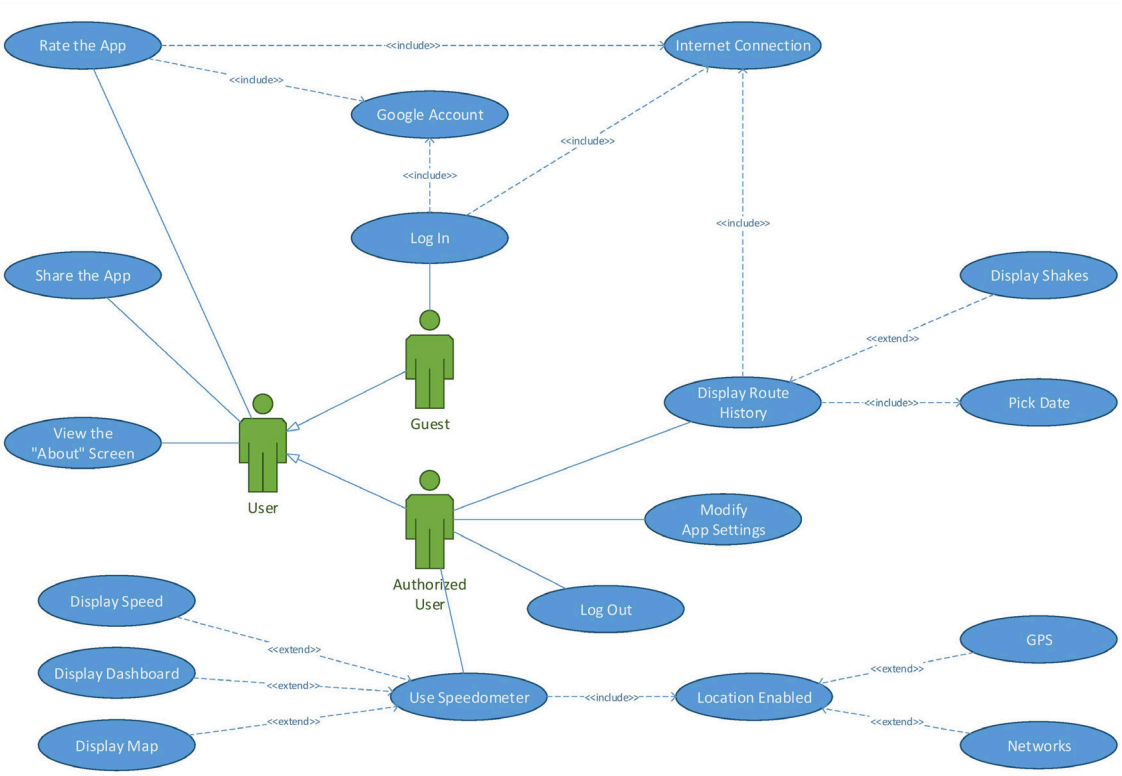
1) **Έναρξη (Inception):** Καθορίζει την προοπτική του έργου.
2) **Εκπόνηση μελέτης (Elaboration):** Σχεδιασμός των απαιτούμενων δραστηριοτήτων και πόρων. Καθορισμός των χαρακτηριστικών και σχεδιασμός της αρχιτεκτονικής.
3) **Κατασκευή (Construction):** Ανάπτυξη του προϊόντος σε μία σειρά βηματικών επαναλήψεων.



Διάγραμμα 1 Φάση «Έναρξη» ~ Διάγραμμα Περιπτώσεων Χρήσης: Χαρακτηριστικά Εφαρμογής.



Διάγραμμα 2 Φάση «Εκπόνηση Μελέτης» ~ Διάγραμμα Περιπτώσεων Χρήσης: Χαρακτηριστικά Εφαρμογής.



Διάγραμμα 3 Φάση «Κατασκευή» ~ Διάγραμμα Περιπτώσεων Χρήσης: Χαρακτηριστικά Εφαρμογής.

ΚΕΦΑΛΑΙΟ 3: Εγχειρίδιο Χρήστη

3.1 Σύντομη Παρουσίαση της Εφαρμογής

Κατά την εκκίνηση της εφαρμογής **Ταχύμετρο (Speedometer)**, μέσω παρασκηνιακού ελέγχου, διαπιστώνεται εάν ο χρήστης της εφαρμογής είναι:

- επισκέπτης ή
- πιστοποιημένος χρήστης.

Κάθε φορά που χρησιμοποιεί την εφαρμογή ένας πιστοποιημένος χρήστης, τότε παραμένει συνδεδεμένος (ακόμα και μετά την επανεκκίνηση της κινητής του συσκευής), μέχρι:

- να επιλέξει ο ίδιος να αποσυνδεθεί ή
- να κάνει εκκαθάριση των δεδομένων της εφαρμογής μέσα από τις ρυθμίσεις της συσκευής ή
- να απεγκαταστήσει την εφαρμογή.

Μέσω του φιλικού περιβάλλοντος που προσφέρεται, ο χρήστης μπορεί να περιηγηθεί στην εφαρμογή και να εκτελέσει τις διαθέσιμες λειτουργίες χωρίς δυσκολία.

Η εφαρμογή «Ταχύμετρο» (Speedometer) διατίθεται ελεύθερα μέσω του Play Store στην ακόλουθη διεύθυνση:

https://play.google.com/store/apps/details?id=net.bplaced.esigala1.speedometer

Εναλλακτικά, μπορείτε να σαρώσετε τον κωδικό QR που βρίσκεται πιο κάτω, μέσω της κινητής σας συσκευής, ώστε να μεταβείτε απευθείας στην σελίδα του Ταχύμετρου στο Play Store:

Στη συνέχεια, παρουσιάζονται αναλυτικά οι λειτουργίες της εφαρμογής, με τα κατάλληλα στιγμιότυπα οθόνης (screenshots).

3.2 Παρουσίαση Σεναρίων Λειτουργίας

Για την κατανόηση των βημάτων που πρέπει να ακολουθήσει ένας χρήστης, ούτως ώστε να εκτελέσει κάποιες λειτουργίες, χωρίς να χάσει χρόνο σε άσκοπη περιήγηση μέσα στην εφαρμογή, χρησιμοποιούμε τα **Σενάρια Λειτουργίας**. Τα βήματα παρουσιάζονται μέσω στιγμιότυπων οθόνης (screenshots), τα οποία είτε αφορούν διαφορετικές οθόνες της εφαρμογής είτε διαφορετικές ενέργειες που πρέπει να εκτελέσει ο χρήστης.

Τα χαρακτηριστικά της κινητής συσκευής, που χρησιμοποιήθηκε για τη λήψη των στιγμιότυπων οθόνης, είναι:

- ✓ **Μοντέλο:** LG G2 (D802)
- ✓ **Διαστάσεις Οθόνης:** 5.2 inches
- ✓ **Ανάλυση Οθόνης:** 1080 x 1920 pixels
- ✓ **Λειτουργικό:** Android 5.0.2 (Lollipop – Επίπεδο API 21)

Επιπρόσθετα χαρακτηριστικά της οθόνης της συσκευής αυτής παρατίθενται στην εικόνα που ακολουθεί:

Device	Platform	Screen dimensions in cm	Aspect Ratio	Width × Height dp	Width × Height px	Density
LG G2	Android	5.2 in 2.5 × 4.5 in	16 : 9	360 × 640 dp	1080 × 1920 px	3.0 xxhdpi

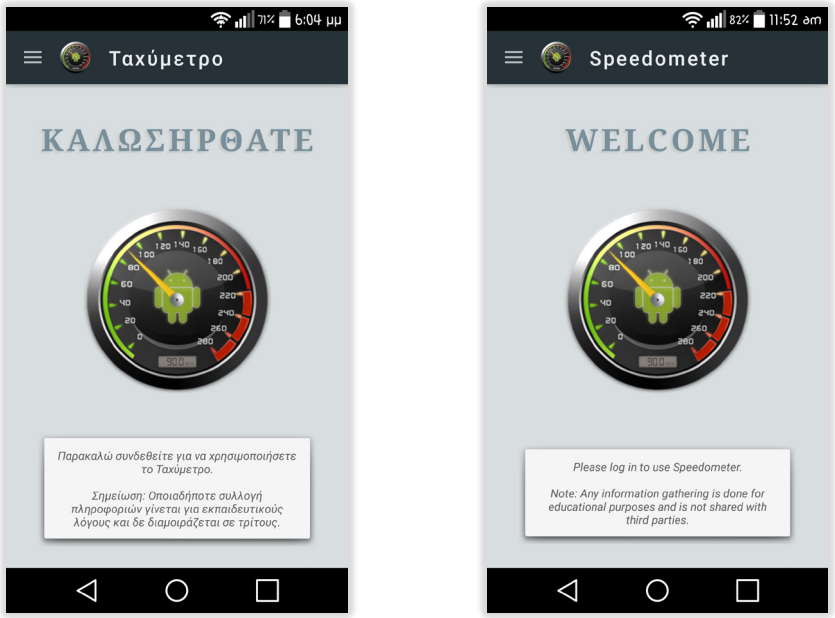
Εικόνα 37 Χαρακτηριστικά Οθόνης LG G2 *(Πηγή: Device Metrics ~ https://material.io/devices/)*.

3.2.1 Αρχική Οθόνη Χρήστη

Η **Αρχική Οθόνη (Splash Screen)** εμφανίζεται κατά την εκκίνηση της εφαρμογής, ανεξάρτητα από τον τύπο του χρήστη.

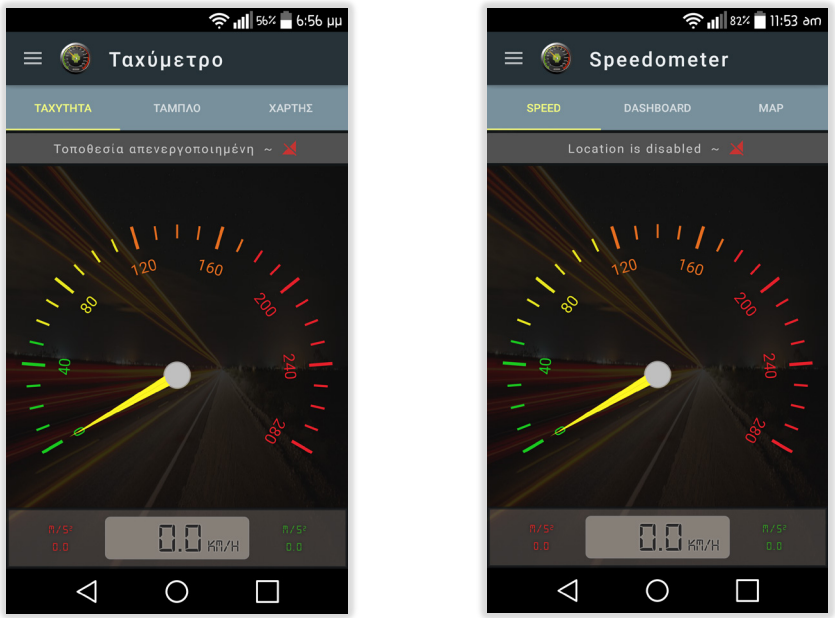
Εικόνα 38 Αρχική οθόνη για όλους τους τύπους χρηστών.

Στη συνέχεια γίνεται παρασκηνιακός έλεγχος για το αν υπάρχει πιστοποιημένος χρήστη ή όχι. Στην περίπτωση που δε βρεθεί πιστοποιημένος χρήστης, τότε η εφαρμογή μεταβαίνει στην **Αρχική Οθόνη του Επισκέπτη**:



Εικόνα 39 Αρχική οθόνη επισκέπτη.

Σε αντίθετη περίπτωση, μεταβαίνει στην **Αρχική Οθόνη του Πιστοποιημένου Χρήστη**:

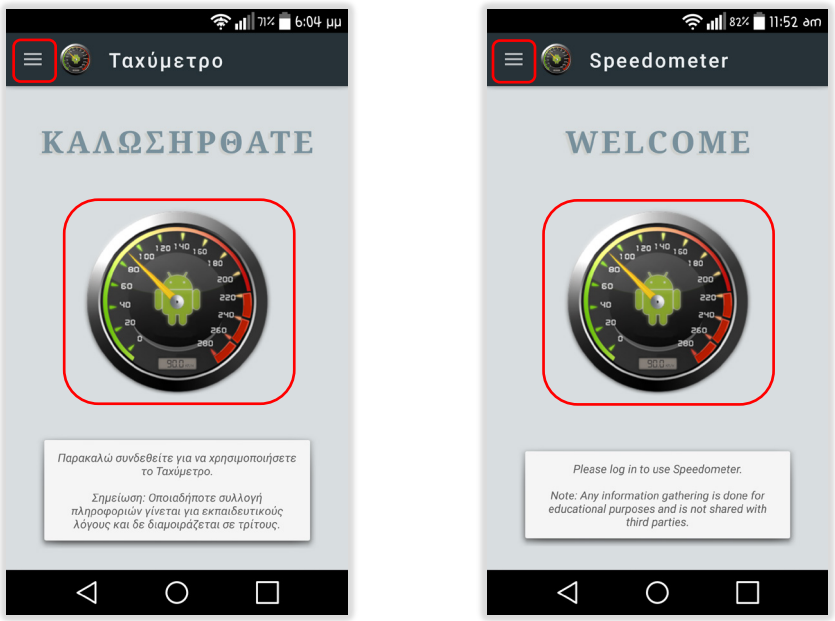


Εικόνα 40 Αρχική οθόνη πιστοποιημένου χρήστη.

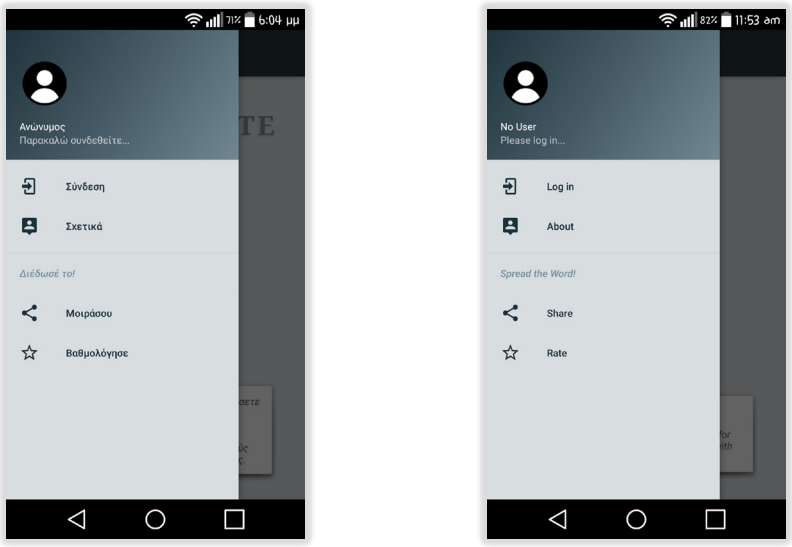
3.2.2 Εμφάνιση Συρταριού Πλοήγησης

Για να μπορέσει ο χρήστης να μεταβεί σε διαφορετικές οθόνες της εφαρμογής, υπάρχει το πλευρικό **Συρτάρι Πλοήγησης** (πάνελ που εμφανίζεται στο αριστερό άκρο της οθόνης) και προσαρμόζεται κάθε φορά στον τύπο χρήστη.

- Οι επισκέπτες μπορεί να εμφανίσουν το συρτάρι πλοήγησης με δύο τρόπους:
  - πατώντας πάνω στο εικονίδιο Hamburger, στην μπάρα της εφαρμογής, ή
  - πατώντας πάνω στην εικόνα στο κέντρο της οθόνης, με το λογότυπο της εφαρμογής:

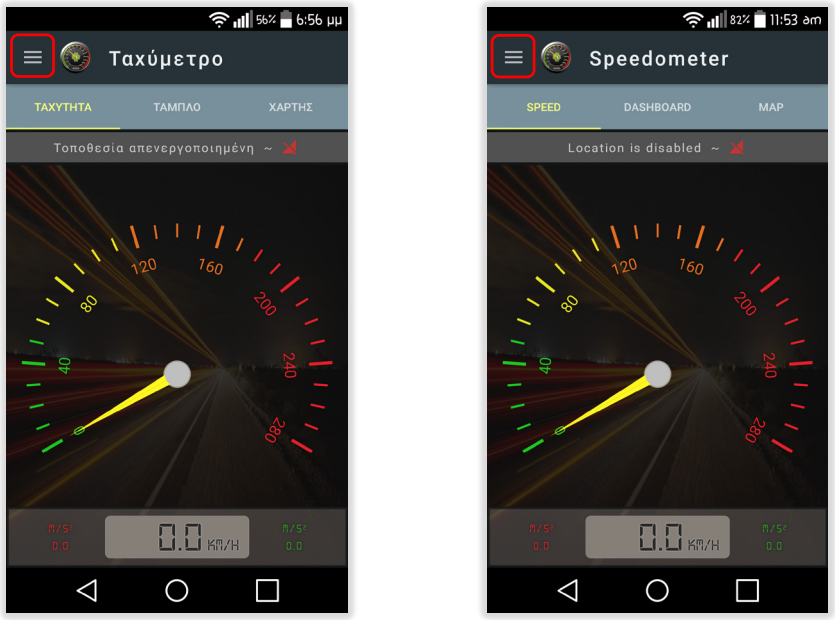


Εικόνα 41 Μέθοδοι εμφάνισης του συρταριού πλοήγησης για τον επισκέπτη.

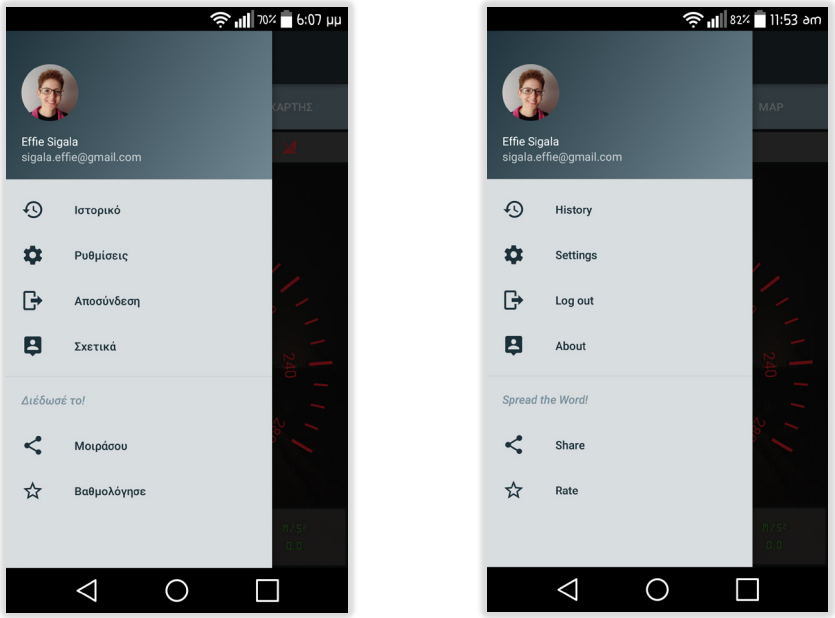


Εικόνα 42 Επιλογές πλοήγησης του επισκέπτη.

❖ Οι πιστοποιημένοι χρήστες μπορούν να εμφανίσουν το συρτάρι πλοήγησης πατώντας πάνω στο εικονίδιο Hamburger, που βρίσκεται στην μπάρα της εφαρμογής:



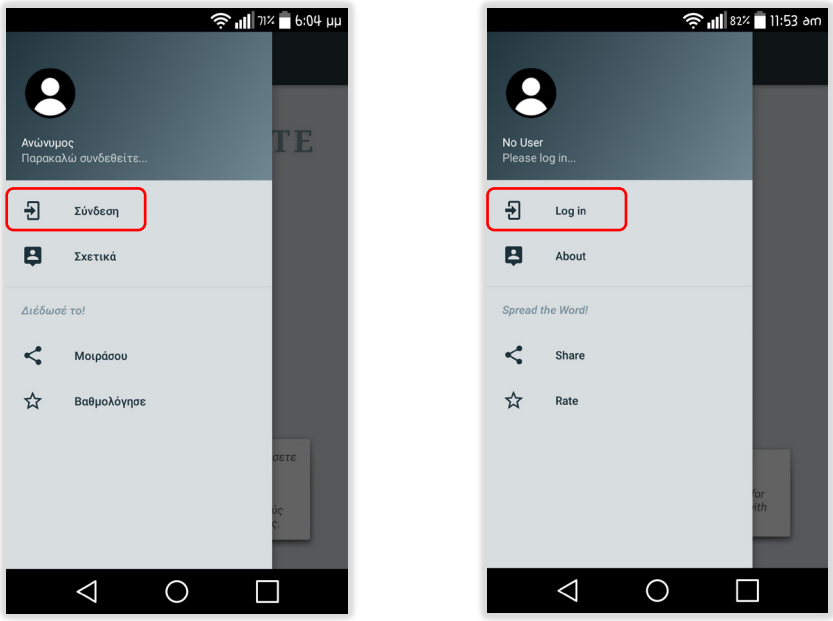
Εικόνα 43 Μέθοδος εμφάνισης του συρταριού πλοήγησης για τον πιστοποιημένο χρήστη.



Εικόνα 44 Επιλογές πλοήγησης του πιστοποιημένου χρήστη.

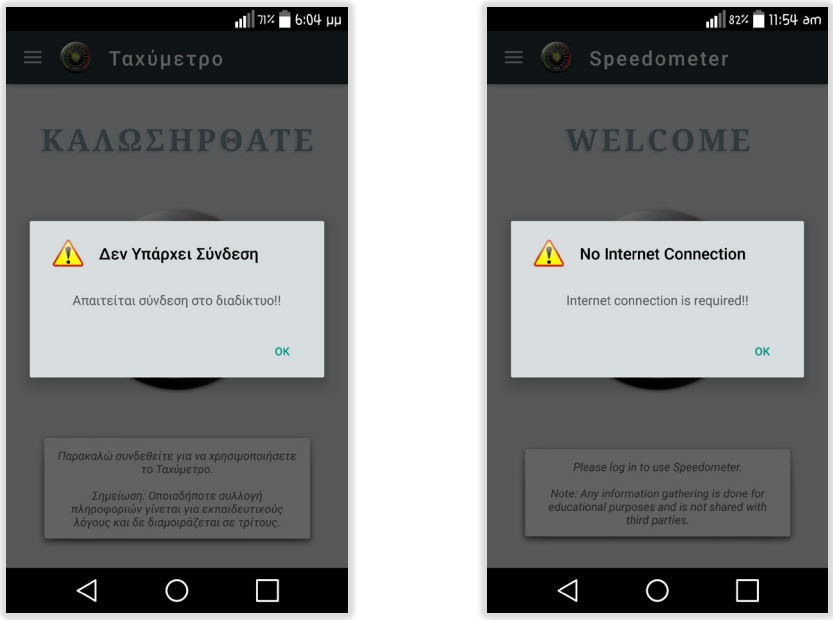
3.2.3 Σύνδεση Χρήστη

Ο επισκέπτης, δηλαδή ο μη πιστοποιημένος χρήστης, μπορεί να συνδεθεί στην εφαρμογή, επιλέγοντας «Σύνδεση» από το συρτάρι πλοήγησης:



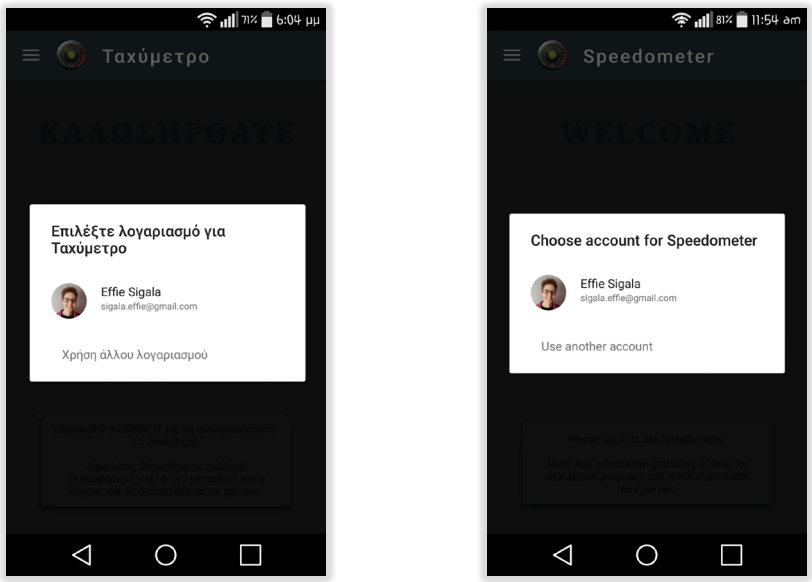
Εικόνα 45 Επιλογή «Σύνδεση» από το συρτάρι πλοήγησης του επισκέπτη.

Στην περίπτωση που δεν υπάρχει σύνδεση στο διαδίκτυο, τότε εμφανίζεται το ακόλουθο μήνυμα:



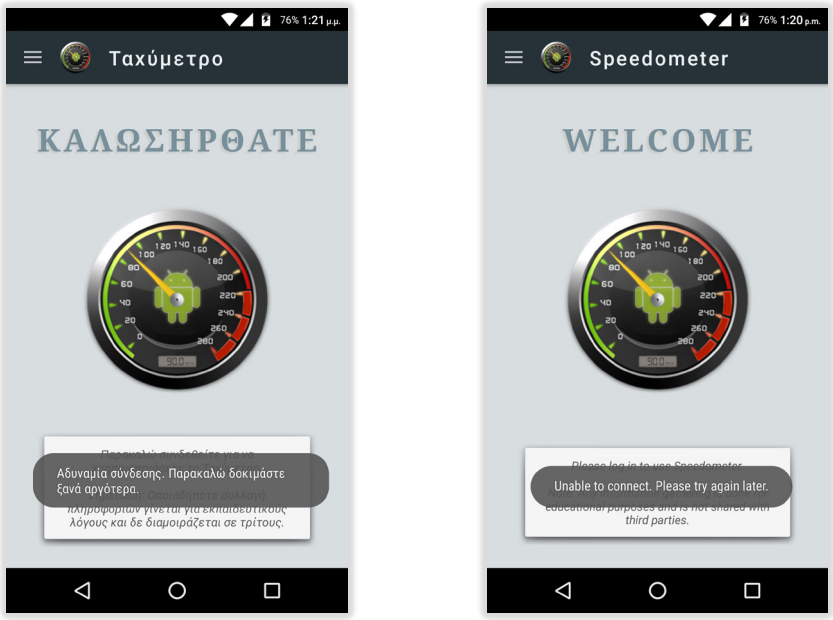
Εικόνα 46 Μήνυμα ενημέρωσης του χρήστη για την ανάγκη σύνδεσης στο διαδίκτυο.

Εάν υπάρχει σύνδεση στο διαδίκτυο, τότε εμφανίζεται ένα παράθυρο διαλόγου με όλους τους διαθέσιμους λογαριασμούς Google που έχει συνδέσει ο χρήστης στη συσκευή, δηλαδή όλα τα email που χρησιμοποιεί για την είσοδό του στο Play Store. Εάν έχει μόνο ένα email τότε εμφανίζεται μόνο αυτό:



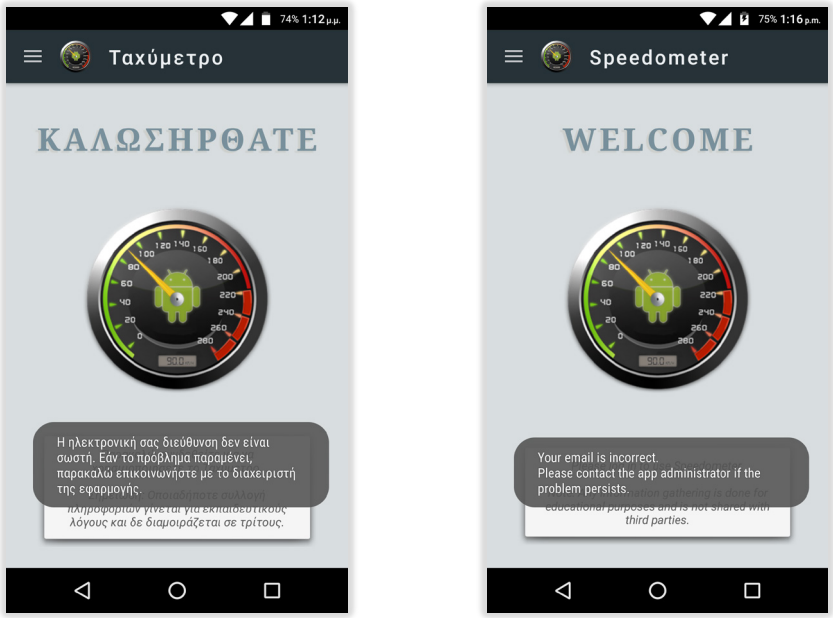
Εικόνα 47 Παράθυρο διαλόγου με όλους τους διαθέσιμους λογαριασμούς Google, για τη σύνδεση του χρήστη στην εφαρμογή.

Στην περίπτωση που ο χρήστης αλλάξει γνώμη και δε θέλει να συνδεθεί στην εφαρμογή ή προκύψει κάποιο συστημικό σφάλμα κατά τη διαδικασία πιστοποίησής του, τότε εμφανίζεται το ακόλουθο μήνυμα σφάλματος και επιστρέφει στην αρχική οθόνη επισκέπτη:



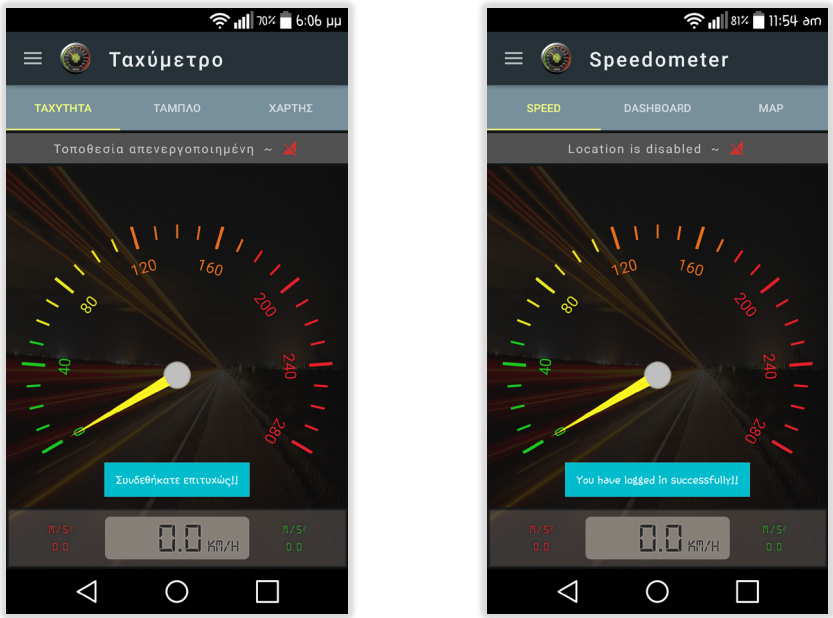
Εικόνα 48 Μήνυμα ανεπιτυχούς σύνδεσης του χρήστη.

Στην περίπτωση που ο χρήστης επιλέξει email, το οποίο έχει αποκλειστεί από το διαχειριστή της εφαρμογής (μέσω του Firebase Authentication), τότε εμφανίζεται το ακόλουθο μήνυμα σφάλματος και επιστρέφει στην αρχική οθόνη επισκέπτη:



Εικόνα 49 Μήνυμα ανεπιτυχούς πιστοποίησης του χρήστη, λόγω αποκλεισμού του από το διαχειριστή.

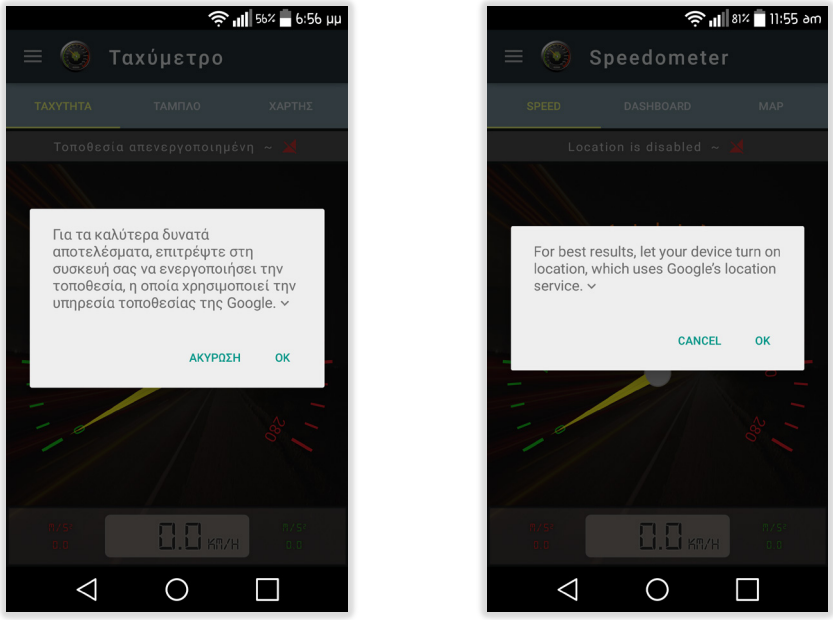
Σε οποιαδήποτε άλλη περίπτωση, γίνεται πιστοποίησή του email που επέλεξε ο χρήστης μέσω του **Identity Toolkit της Google** και εμφανίζεται το ακόλουθο μήνυμα επιτυχούς σύνδεσης, μεταβαίνοντας στην αρχική οθόνη του πιστοποιημένου χρήστη:



Εικόνα 50 Μήνυμα επιτυχούς σύνδεσης του χρήστη στην εφαρμογή.

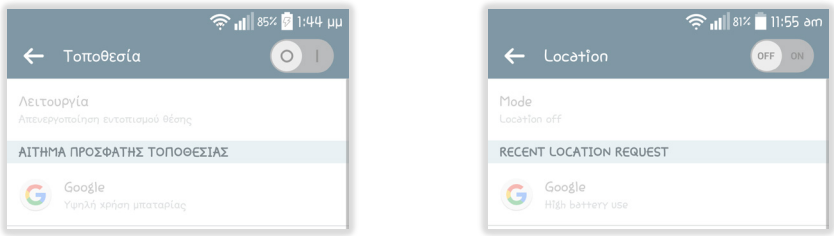
3.2.4 Ενεργοποίηση Τοποθεσίας

Κάθε φορά που γίνεται εκκίνηση της εφαρμογής και εντοπίζεται πιστοποιημένος χρήστης, αλλά η **Τοποθεσία** της συσκευής είναι απενεργοποιημένη, εμφανίζεται το κάτωθι παράθυρο διαλόγου για την αυτόματη ενεργοποίησή της από το σύστημα:

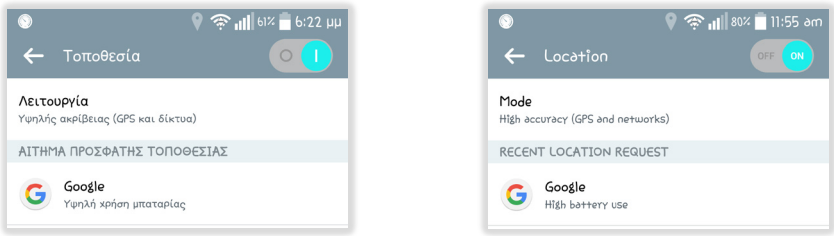


Εικόνα 51 Παράθυρο διαλόγου ενεργοποίησης τοποθεσίας.

❖ Εναλλακτικά, η ενεργοποίηση της τοποθεσίας γίνεται μέσα από τις ρυθμίσεις της συσκευής:

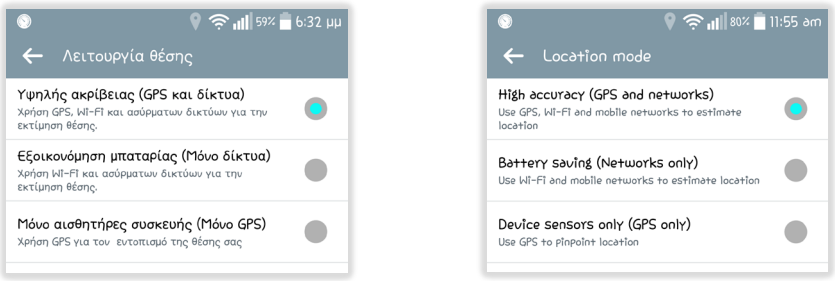


Εικόνα 52 Τοποθεσία συσκευής απενεργοποιημένη.

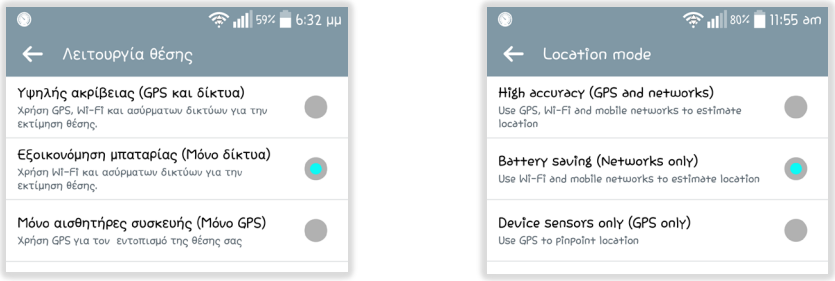


Εικόνα 53 Τοποθεσία συσκευής ενεργοποιημένη.

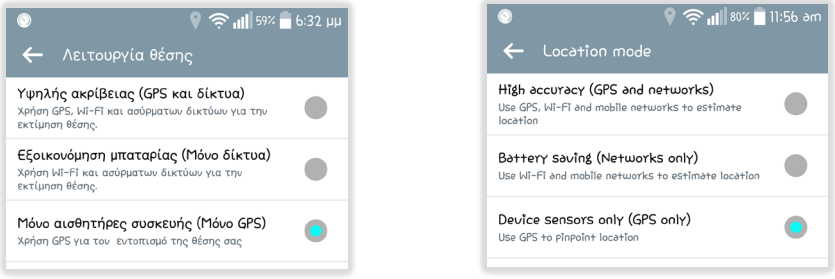
Κάθε κινητή συσκευή που διαθέτει ενσωματωμένο GPS, προσφέρει τρεις διαφορετικές μεθόδους λήψης της τοποθεσίας, όπως αυτές παρουσιάζονται στα στιγμιότυπα οθόνης που ακολουθούν:



Εικόνα 54 Λήψη τοποθεσίας μέσω GPS και δικτύων.



Εικόνα 55 Λήψη τοποθεσίας μέσω δικτύων.



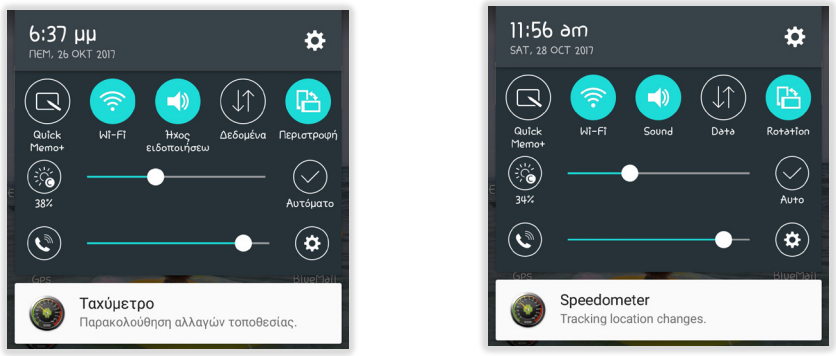
Εικόνα 56 Λήψη τοποθεσίας μέσω GPS.

Σημείωση: Στην περίπτωση που μία κινητή συσκευή δε διαθέτει GPS, τότε η λήψη τοποθεσίας γίνεται αποκλειστικά μέσω των δικτύων της συσκευής.

- ❖ Όταν η τοποθεσία είναι ενεργοποιημένη και η εφαρμογή λειτουργεί, τότε:
  - ✓ Εμφανίζεται ένα γραφικό στοιχείο στη γραμμή κατάστασης (status bar):

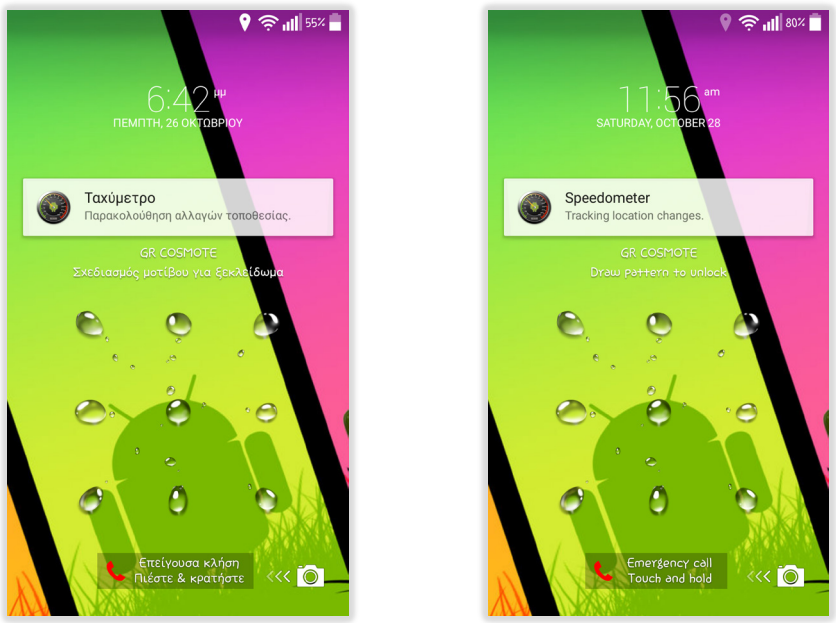
Εικόνα 57 Εικονίδιο εφαρμογής στη γραμμή κατάστασης.

✓ Εμφανίζεται μία ειδοποίηση (notification) όταν η οθόνη είναι ξεκλείδωτη:



Εικόνα 58 Ειδοποίηση εφαρμογής όταν η οθόνη είναι ξεκλείδωτη.

✓ Εμφανίζεται μία ειδοποίηση (notification) όταν η οθόνη είναι κλειδωμένη:

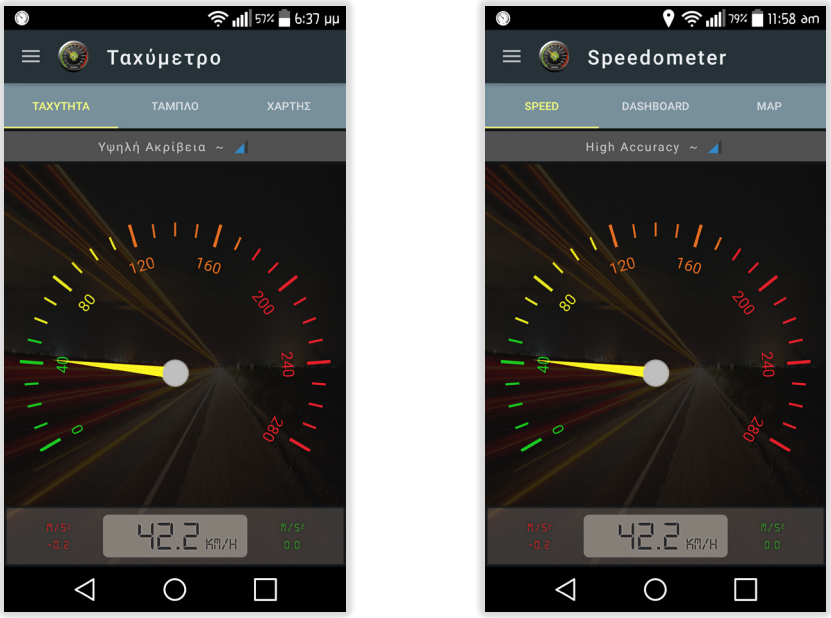


Εικόνα 59 Ειδοποίηση εφαρμογής όταν η οθόνη είναι κλειδωμένη.

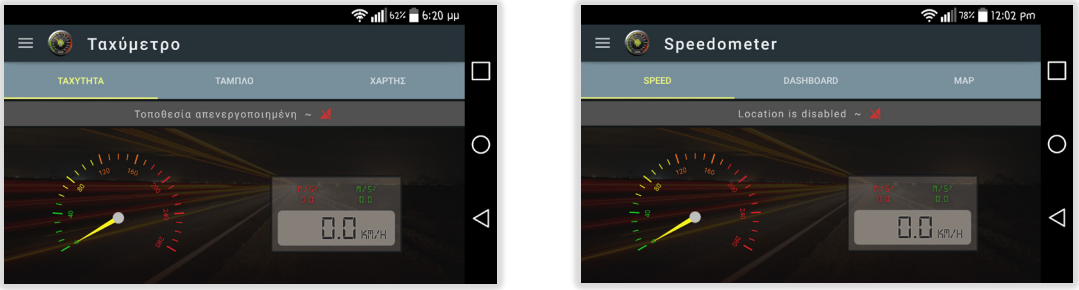
3.2.5 Καρτέλες Πιστοποιημένου Χρήστη

Ο πιστοποιημένος χρήστης, ενώ βρίσκεται στην αρχική του οθόνη, μπορεί να πλοηγηθεί μεταξύ τριών διαφορετικών καρτελών, με χρήση της οριζόντιας χειρονομίας δακτύλων (αριστερά - δεξιά), ή επιλέγοντας πάνω στον τίτλο της καρτέλας που επιθυμεί να προβάλλει. Οι διαθέσιμες καρτέλες είναι οι ακόλουθες τρεις:

- **Καρτέλα Ταχύτητα**, που περιλαμβάνει:
  - Μπάρα ενημερώσεων της κατάστασης και της ακρίβειας του γεωγραφικού στίγματος.
  - Προβολή της ταχύτητας μέσω αναλογικού οργάνου.
  - Προβολή της ταχύτητας μέσω ψηφιακού οργάνου.
  - Προβολή της αρνητικής επιτάχυνσης με κόκκινο χρώμα και της θετικής επιτάχυνσης με πράσινο χρώμα.



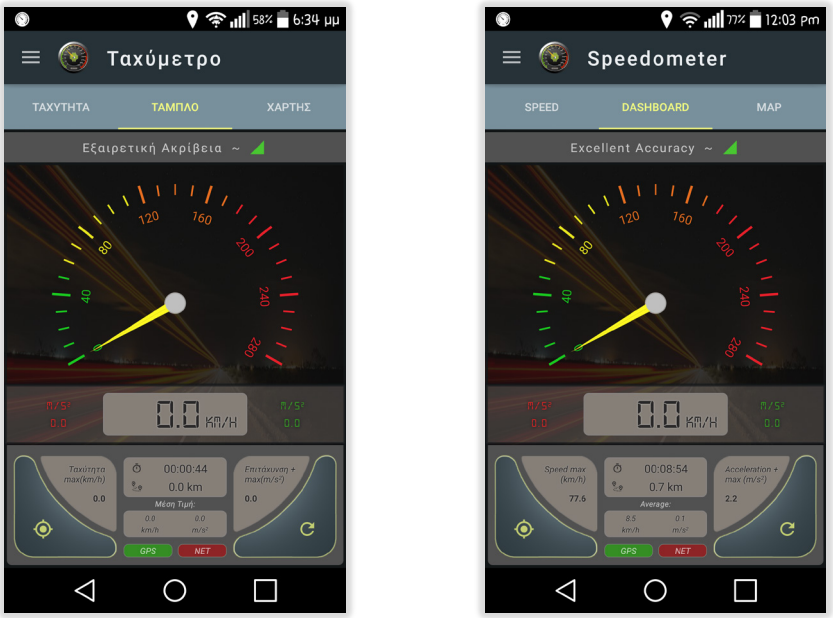
Εικόνα 60 Καρτέλα πιστοποιημένου χρήστη: «Ταχύτητα» (κατακόρυφη διάταξη).



Εικόνα 61 Καρτέλα πιστοποιημένου χρήστη: «Ταχύτητα» (οριζόντια διάταξη).

- **Καρτέλα Ταμπλό**, που περιλαμβάνει:
  - Μπάρα ενημερώσεων της κατάστασης και της ακρίβειας του γεωγραφικού στίγματος.
  - Προβολή της ταχύτητας μέσω αναλογικού οργάνου.

- Προβολή της ταχύτητας μέσω ψηφιακού οργάνου.
- Προβολή της αρνητικής επιτάχυνσης με κόκκινο χρώμα και της θετικής επιτάχυνσης με πράσινο χρώμα.
- Κουμπί μετάβασης στις ρυθμίσεις «Τοποθεσία» της συσκευής.
- Προβολή μέγιστης ταχύτητας.
- Προβολή χρόνου που έχει περάσει από τον τελευταίο μηδενισμό των μετρήσεων.
- Προβολή απόστασης που έχει διανύσει ο χρήστης.
- Προβολή μέσης ταχύτητας.
- Προβολή μέσης επιτάχυνσης.
- Εάν γίνεται χρήση του GPS για τον εντοπισμό της τοποθεσίας, τότε το φόντο της ένδειξης «GPS» γίνεται πράσινο, διαφορετικά κόκκινο.
- Εάν γίνεται χρήση των δικτύων για τον εντοπισμό της τοποθεσίας, τότε το φόντο της ένδειξης «NET» γίνεται πράσινο, διαφορετικά κόκκινο.
- Προβολή μέγιστης επιτάχυνσης.
- Κουμπί μηδενισμού μετρήσεων.



Εικόνα 62 Καρτέλα πιστοποιημένου χρήστη: «Ταμπλό» (κατακόρυφη διάταξη).

Εικόνα 63 Καρτέλα πιστοποιημένου χρήστη: «Ταμπλό» (οριζόντια διάταξη).

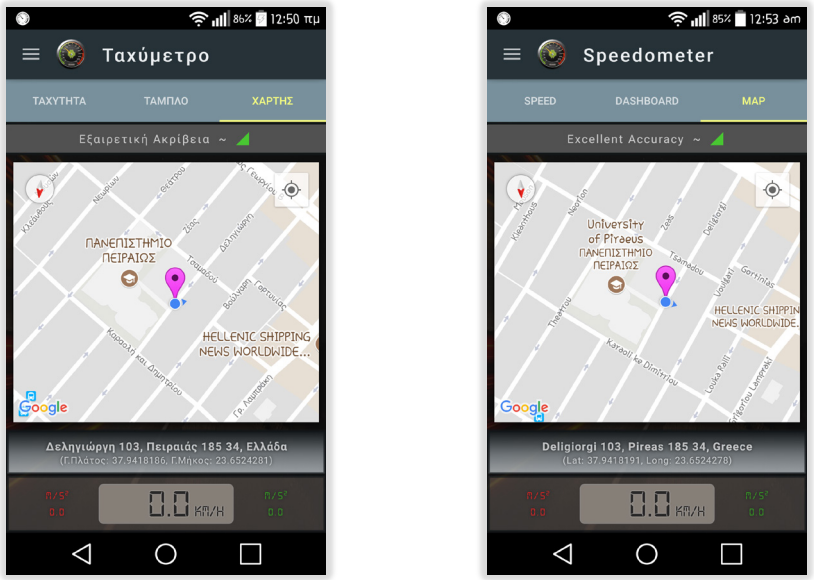
Εάν ο χρήστης πατήσει πάνω στο κουμπί μηδενισμού των μετρήσεων, τότε εμφανίζεται το κάτωθι παράθυρο διαλόγου για την επιβεβαίωση της ενέργειάς του:

Εικόνα 64 Παράθυρο διαλόγου μηδενισμού μετρήσεων.

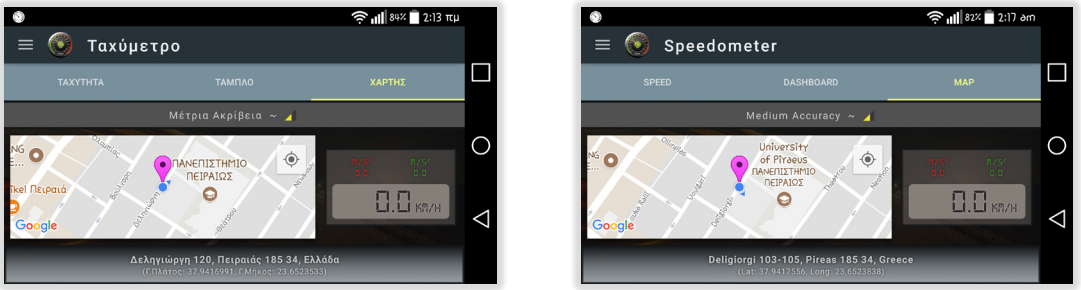
Εάν ο χρήστης ανταποκριθεί θετικά, τότε οι μετρήσεις μηδενίζονται:

Εικόνα 65 Μηδενισμός μετρήσεων.

- **Καρτέλα Χάρτης**, που περιλαμβάνει:
  - Μπάρα ενημερώσεων της κατάστασης και της ακρίβειας του γεωγραφικού στίγματος.
  - Το χάρτη της Google, με αγκυροβολημένο εικονίδιο (marker) στην τρέχουσα τοποθεσία του χρήστη.
  - Την οδό που αντιστοιχεί στην τρέχουσα τοποθεσία του χρήστη.
  - Προβολή της ταχύτητας μέσω ψηφιακού οργάνου.
  - Προβολή της αρνητικής επιτάχυνσης με κόκκινο χρώμα και της θετικής επιτάχυνσης με πράσινο χρώμα.



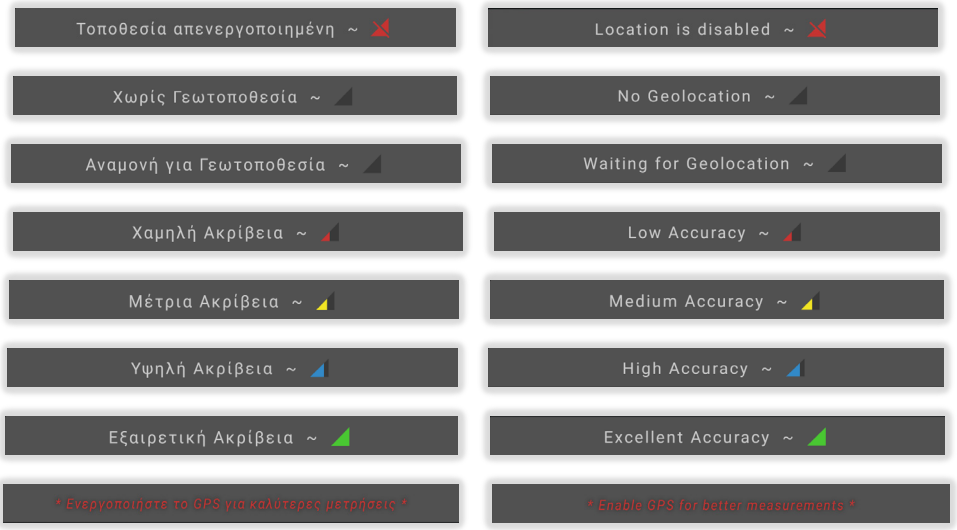
Εικόνα 66 Καρτέλα πιστοποιημένου χρήστη: «Χάρτης» (κατακόρυφη διάταξη).



Εικόνα 67 Καρτέλα πιστοποιημένου χρήστη: «Χάρτης» (οριζόντια διάταξη).

Η μπάρα ενημερώσεων προβάλει μηνύματα στο χρήστη σχετικά με τη δυνατότητα ή μη της λήψης του γεωγραφικού στίγματος, καθώς και το πόσο ακριβές ή όχι είναι το στίγμα που έχει ληφθεί. Τα διαθέσιμα ενημερωτικά μηνύματα είναι:

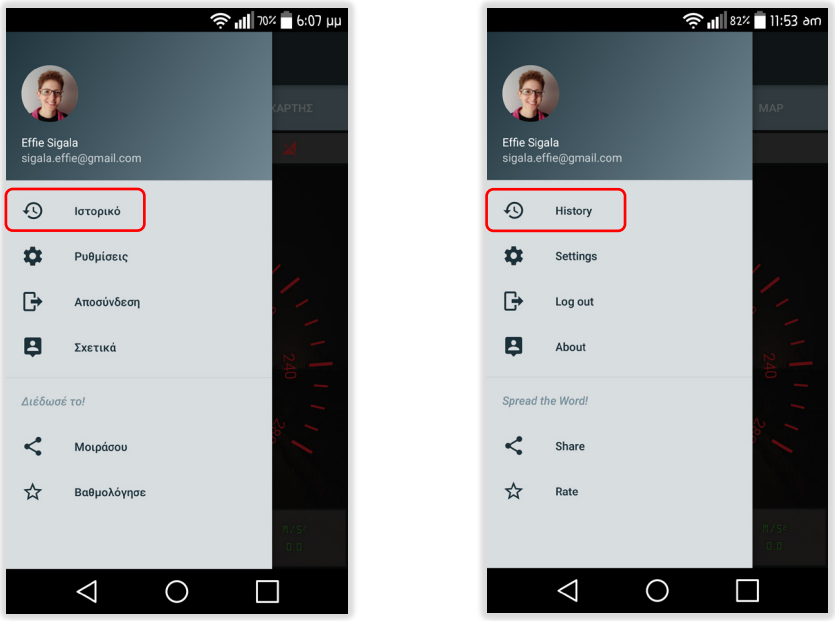
- **Τοποθεσία Απενεργοποιημένη:** Ο εντοπισμός της τοποθεσίας είναι απενεργοποιημένος από τη συσκευή.
- **Χωρίς Γεωτοποθεσία:** Ο εντοπισμός της τοποθεσίας είναι ενεργοποιημένος από τη συσκευή, αλλά δεν έχει ληφθεί ακόμα η τελευταία γνωστή τοποθεσία του χρήστη.
- **Αναμονή για Γεωτοποθεσία:** Έχει ληφθεί η τελευταία γνωστή τοποθεσία του χρήστη, αλλά δεν έχει εντοπιστεί ακόμα κάποια αλλαγή στην τρέχουσα τοποθεσία του.
- **Χαμηλή Ακρίβεια:** Η εφαρμογή λαμβάνει την τρέχουσα τοποθεσία του χρήστη, με χαμηλή ακρίβεια του γεωγραφικού στίγματος.
- **Μέτρια Ακρίβεια:** Η εφαρμογή λαμβάνει την τρέχουσα τοποθεσία του χρήστη, με μέτρια ακρίβεια του γεωγραφικού στίγματος.
- **Υψηλή Ακρίβεια:** Η εφαρμογή λαμβάνει την τρέχουσα τοποθεσία του χρήστη, με υψηλή ακρίβεια του γεωγραφικού στίγματος.
- **Εξαιρετική Ακρίβεια:** Η εφαρμογή λαμβάνει την τρέχουσα τοποθεσία του χρήστη, με εξαιρετική ακρίβεια του γεωγραφικού στίγματος.
- **Ενεργοποιήστε το GPS για καλύτερες μετρήσεις:** Ο εντοπισμός της τοποθεσίας είναι ενεργοποιημένος, αλλά λαμβάνει ενημερώσεις μόνο από τα δίκτυα και όχι από το GPS.



Εικόνα 68 Ενημερωτικά μηνύματα κατάστασης και ακρίβειας γεωγραφικού στίγματος.

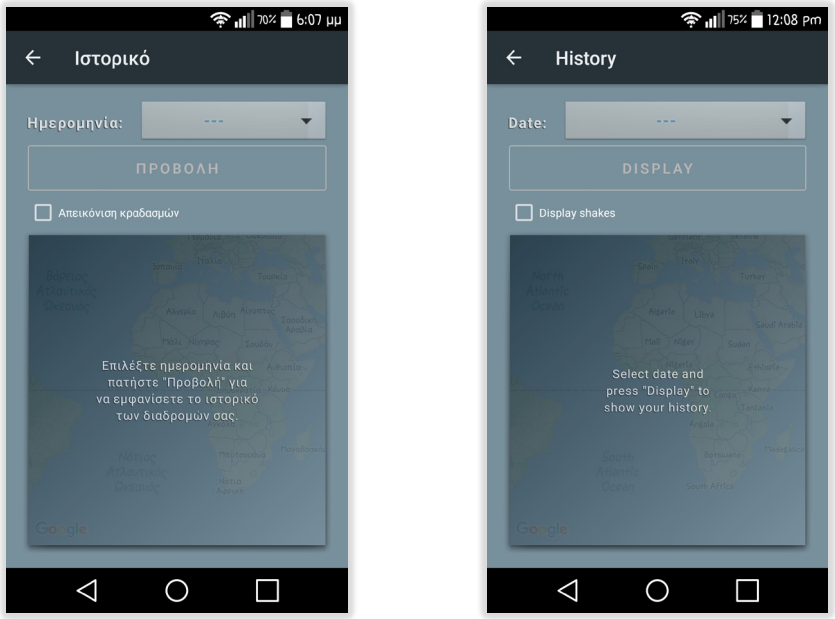
3.2.6 Προβολή Ιστορικού

Ο πιστοποιημένος χρήστης μπορεί να δει τις διαδρομές που έχει διανύσει στο παρελθόν, επιλέγοντας «Ιστορικό» από το συρτάρι πλοήγησης:



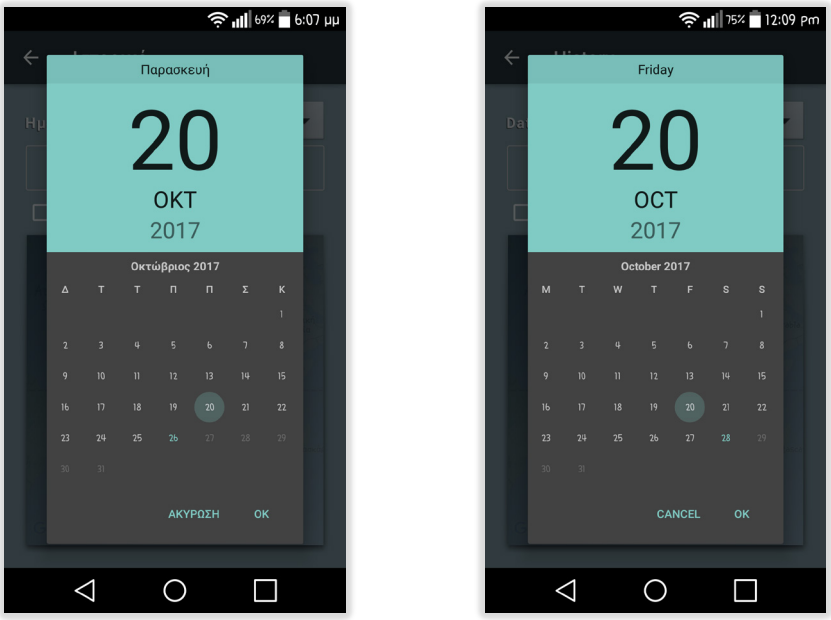
Εικόνα 69 Επιλογή «Ιστορικό» από το συρτάρι πλοήγησης του πιστοποιημένου χρήστη.

Στη συνέχεια, μεταφέρεται στην οθόνη «Ιστορικό», στην οποία καλείται να επιλέξει την ημερομηνία, για την οποία επιθυμεί να εμφανίσει τη διαδρομή που έχει διανύσει. Το κουμπί «ΠΡΟΒΟΛΗ» παραμένει απενεργοποιημένο, έως ότου ο χρήστης επιλέξει ημερομηνία:



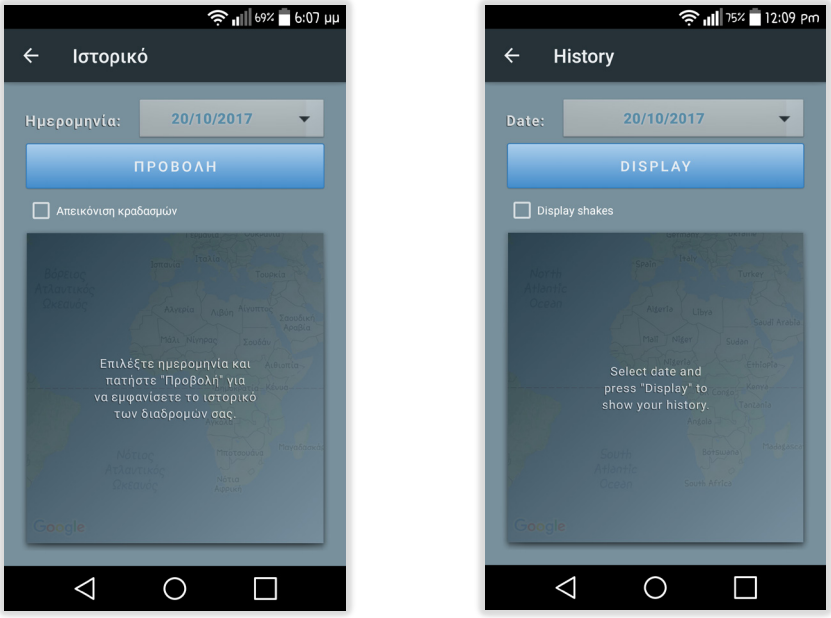
Εικόνα 70 Κουμπί «ΠΡΟΒΟΛΗ» απενεργοποιημένο στην οθόνη «Ιστορικό».

Μέσω ενός παραθύρου διαλόγου ο χρήστης καλείται να επιλέξει την ημερομηνία που επιθυμεί:



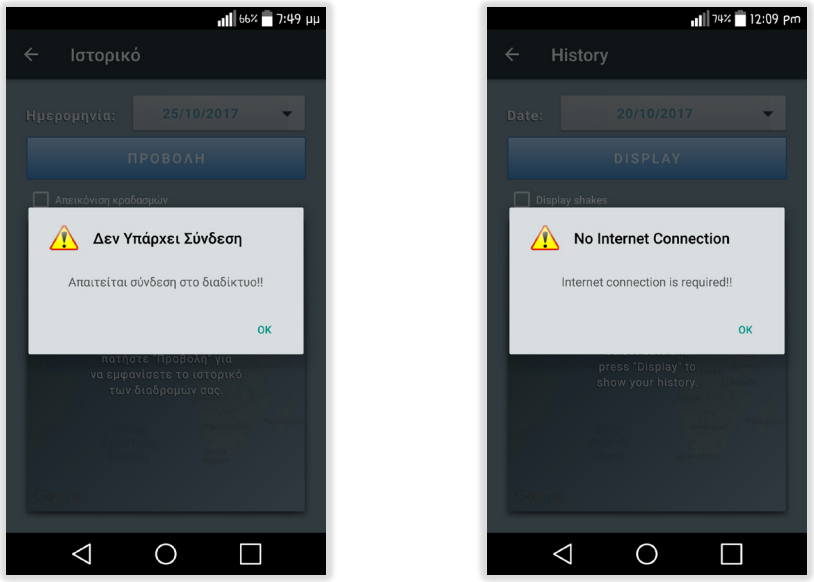
Εικόνα 71 Παράθυρο διαλόγου επιλογής ημερομηνίας.

Μόλις ο χρήστης επιλέξει την ημερομηνία, τότε το παράθυρο διαλόγου κλείνει και το κουμπί «ΠΡΟΒΟΛΗ» ενεργοποιείται:



Εικόνα 72 Κουμπί «ΠΡΟΒΟΛΗ» ενεργοποιημένο στην οθόνη «Ιστορικό».

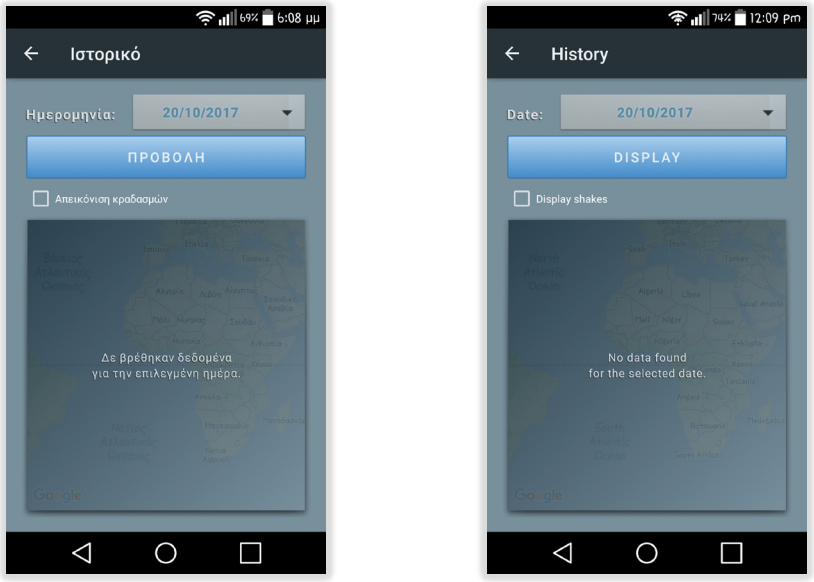
Πατώντας πάνω στο κουμπί «ΠΡΟΒΟΛΗ», εάν δεν υπάρχει σύνδεση στο διαδίκτυο, τότε εμφανίζεται το ακόλουθο μήνυμα:



Εικόνα 73 Μήνυμα ενημέρωσης του χρήστη για την ανάγκη σύνδεσης στο διαδίκτυο.

Εάν υπάρχει σύνδεση στο διαδίκτυο, τότε πατώντας πάνω στο κουμπί «ΠΡΟΒΟΛΗ», γίνεται αναζήτηση στη βάση δεδομένων για καταγραφές που πιθανόν να έχουν γίνει από το συγκεκριμένο χρήστη για τη συγκεκριμένη ημέρα.

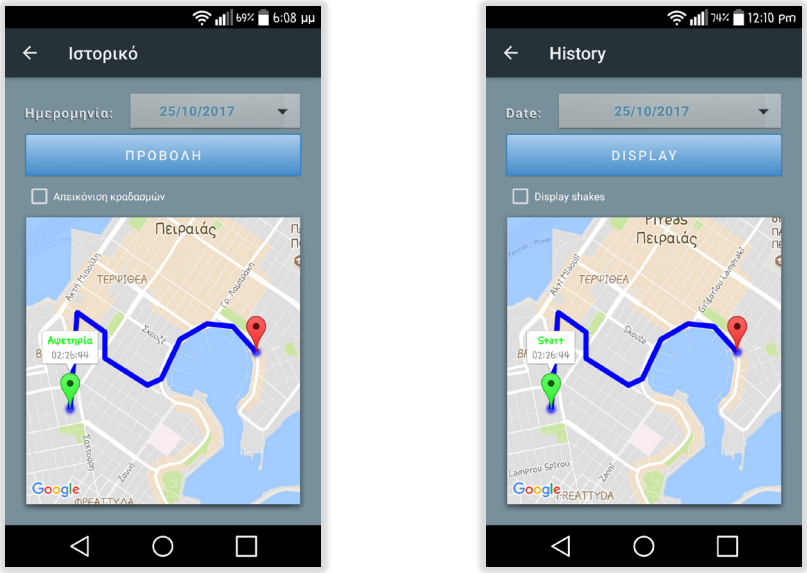
Εάν δεν υπάρχουν καταγραφές, τότε εμφανίζεται το μήνυμα «Δε βρέθηκαν δεδομένα για την επιλεγμένη ημέρα»:



Εικόνα 74 Επιλεγμένη ημέρα στην οθόνη «Ιστορικό», χωρίς καταγεγραμμένα δεδομένα.

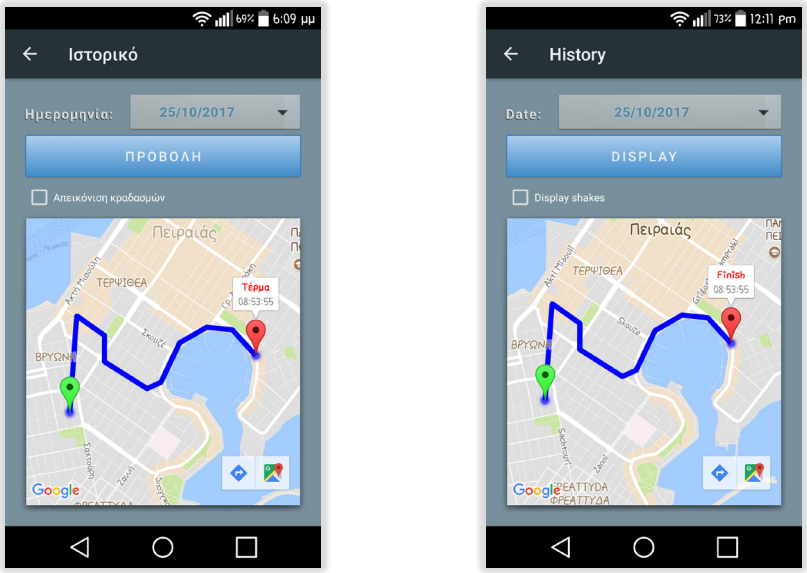
Στην περίπτωση που βρεθούν καταγραφές, τότε εμφανίζεται ο χάρτης της Google και όλα τα γεωγραφικά στίγματα που ανακτήθηκαν από τη βάση δεδομένων, ενώνονται μεταξύ τους με μία λεπτή μπλε γραμμή, αναπαριστώντας τη διαδρομή του χρήστη που καταγράφηκε για τη συγκεκριμένη ημέρα.

Στην αρχή της διαδρομής αγκυροβολείται ένα πράσινο εικονίδιο με την ένδειξη «Αφετηρία» και την ώρα έναρξης της διαδρομής:



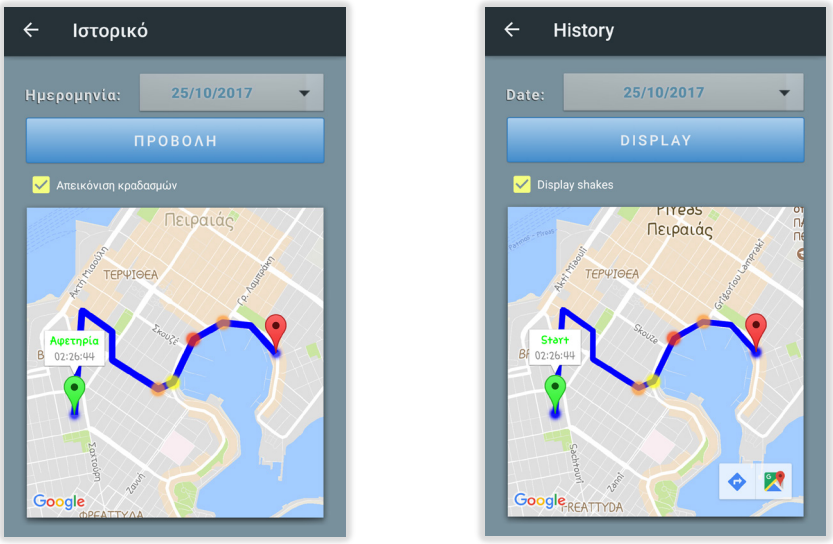
Εικόνα 75 Επιλεγμένη ημέρα στην οθόνη «Ιστορικό», με καταγεγραμμένα δεδομένα.

Στο τέλος της διαδρομής αγκυροβολείται ένα κόκκινο εικονίδιο με την ένδειξη «Τέρμα» και την ώρα λήξης της διαδρομής:



Εικόνα 76 Τέρμα διαδρομής.

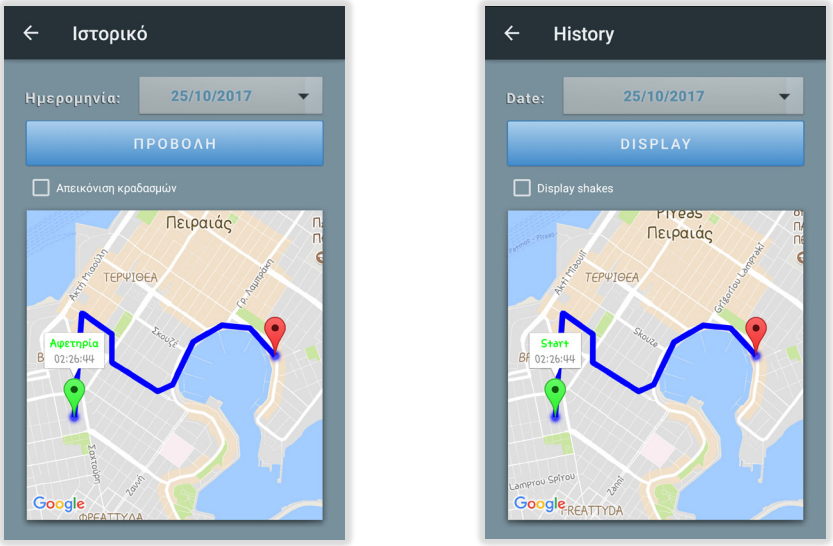
Πατώντας πάνω στο πλαίσιο ελέγχου (checkbox) «Απεικόνιση κραδασμών», εμφανίζονται οι κραδασμοί που έχουν καταγραφεί για τις συγκεκριμένες γεωγραφικές θέσεις:



Εικόνα 77 Προβολή διαδρομής με απεικόνιση κραδασμών.

- Οι κραδασμοί χωρίζονται σε 4 κατηγορίες:
  1) **Μικροί κραδασμοί**, οι οποίοι δεν απεικονίζονται στο χάρτη.
  2) **Μεσαίοι κραδασμοί**, οι οποίοι απεικονίζονται στο χάρτη με κίτρινο χρώμα.
  3) **Μεγάλοι κραδασμοί**, οι οποίοι απεικονίζονται στο χάρτη με πορτοκαλί χρώμα.
  4) **Πολύ μεγάλοι κραδασμοί**, οι οποίοι απεικονίζονται στο χάρτη με κόκκινο χρώμα.

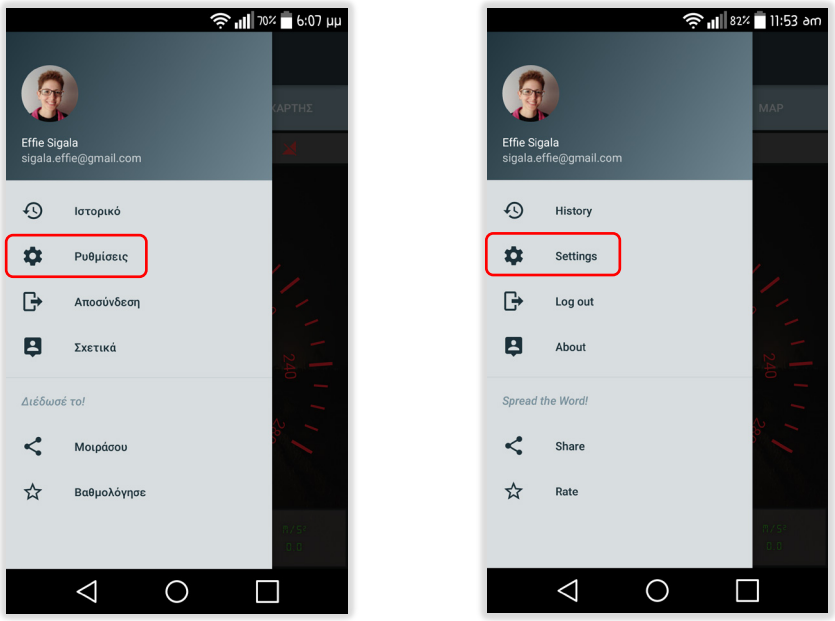
Πατώντας πάλι πάνω στο πλαίσιο ελέγχου «Απεικόνιση κραδασμών», οι κραδασμοί αποκρύπτονται:



Εικόνα 78 Προβολή διαδρομής χωρίς απεικόνιση κραδασμών.

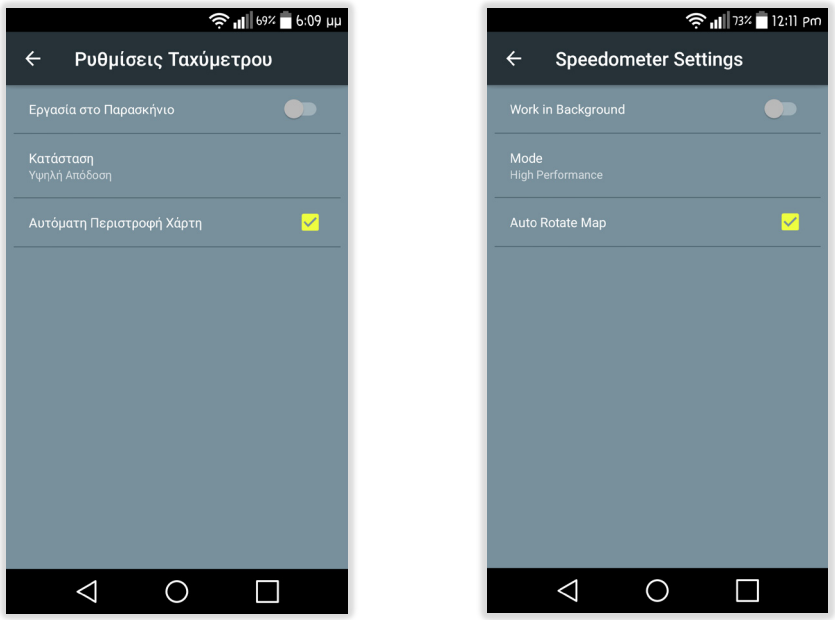
3.2.7 Ρυθμίσεις Εφαρμογής

Ο πιστοποιημένος χρήστης μπορεί να τροποποιήσει τις ρυθμίσεις της εφαρμογής, επιλέγοντας «Ρυθμίσεις» από το συρτάρι πλοήγησης:

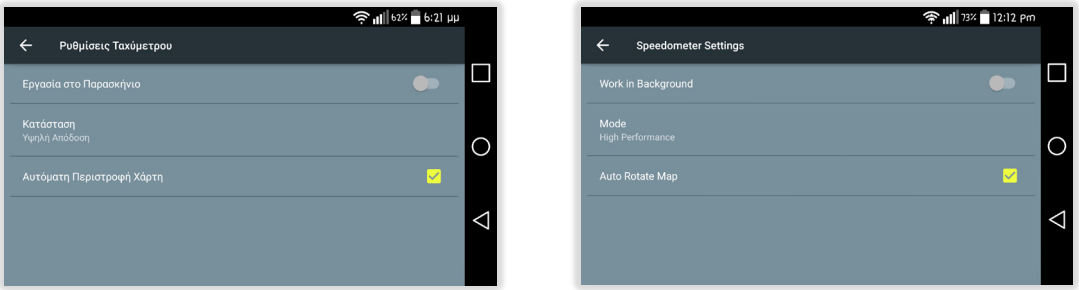


Εικόνα 79 Επιλογή «Ρυθμίσεις» από το συρτάρι πλοήγησης του πιστοποιημένου χρήστη.

Οι προεπιλεγμένες ρυθμίσεις είναι οι ακόλουθες:



Εικόνα 80 Προεπιλεγμένες ρυθμίσεις εφαρμογής (κατακόρυφη διάταξη).



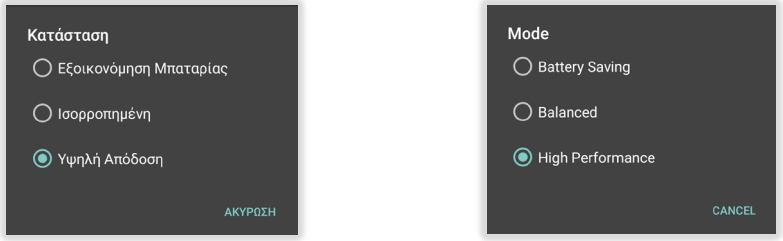
Εικόνα 81 Προεπιλεγμένες ρυθμίσεις εφαρμογής (οριζόντια διάταξη).

Οι διαθέσιμες ρυθμίσεις που μπορούν να τροποποιηθούν είναι:

- **Εργασία στο Παρασκήνιο:** Ο χρήστης μπορεί να επιλέξει εάν θέλει να λειτουργεί η εφαρμογή ενώ βρίσκεται στο παρασκήνιο ή όχι, μέσω διακόπτη (switch). *Προεπιλεγμένη τιμή: Όχι.*

Εικόνα 82 Ρύθμιση «Εργασία στο Παρασκήνιο».

- **Κατάσταση:** Ο χρήστης μπορεί να επιλέξει τη συχνότητα λήψης των αλλαγών τοποθεσίας μέσω πλαισίου διαλόγου, η οποία επηρεάζει την κατανάλωση της μπαταρίας της συσκευής. *Προεπιλεγμένη τιμή: Υψηλή Απόδοση.*



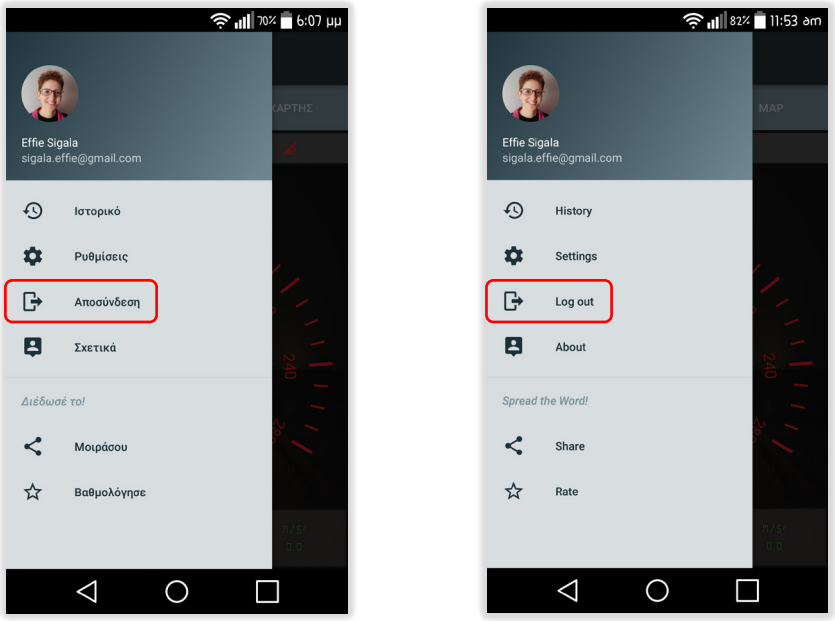
Εικόνα 83 Ρύθμιση «Κατάσταση».

- **Αυτόματη Περιστροφή Χάρτη:** Ο χρήστης μπορεί να επιλέξει εάν ο χάρτης θα περιστρέφεται αυτόματα ή όχι κατά την αλλαγή της τοποθεσίας του, σύμφωνα με την κατεύθυνσή του, μέσω πλαισίου ελέγχου (checkbox). *Προεπιλεγμένη τιμή: Ναι.*

Εικόνα 84 Ρύθμιση «Αυτόματη Περιστροφή Χάρτη».

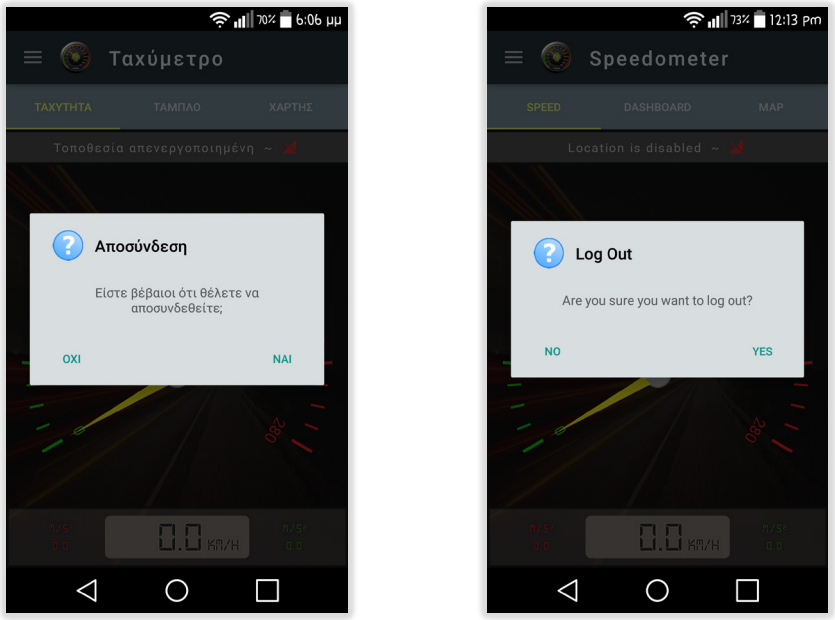
3.2.8 Αποσύνδεση Χρήστη

Ο πιστοποιημένος χρήστης μπορεί να αποσυνδεθεί από την εφαρμογή, επιλέγοντας «Αποσύνδεση» από το συρτάρι πλοήγησης:



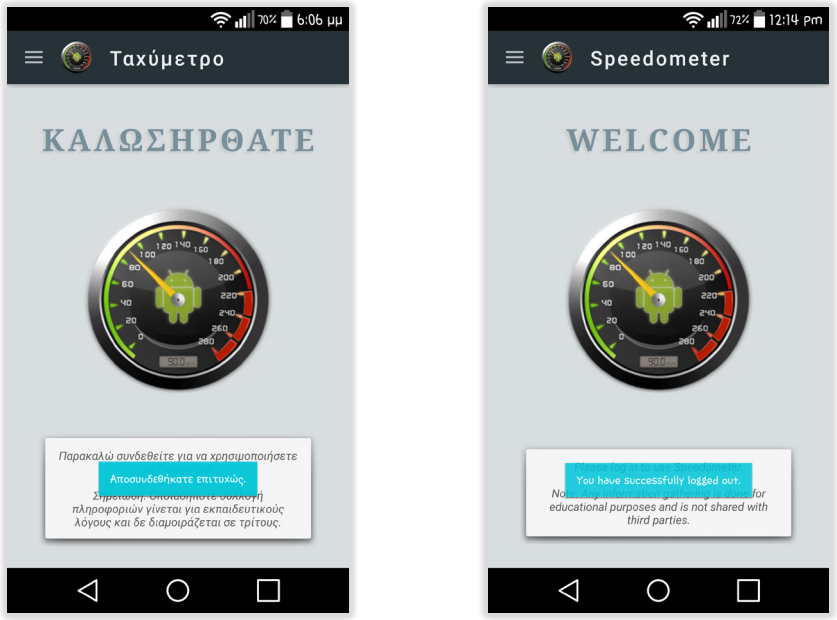
Εικόνα 85 Επιλογή «Αποσύνδεση» από το συρτάρι πλοήγησης του πιστοποιημένου χρήστη.

Στη συνέχεια εμφανίζεται ένα πλαίσιο διαλόγου επιβεβαίωσης της ενέργειας που ζήτησε ο χρήστης:



Εικόνα 86 Πλαίσιο διαλόγου αποσύνδεσης χρήστη.

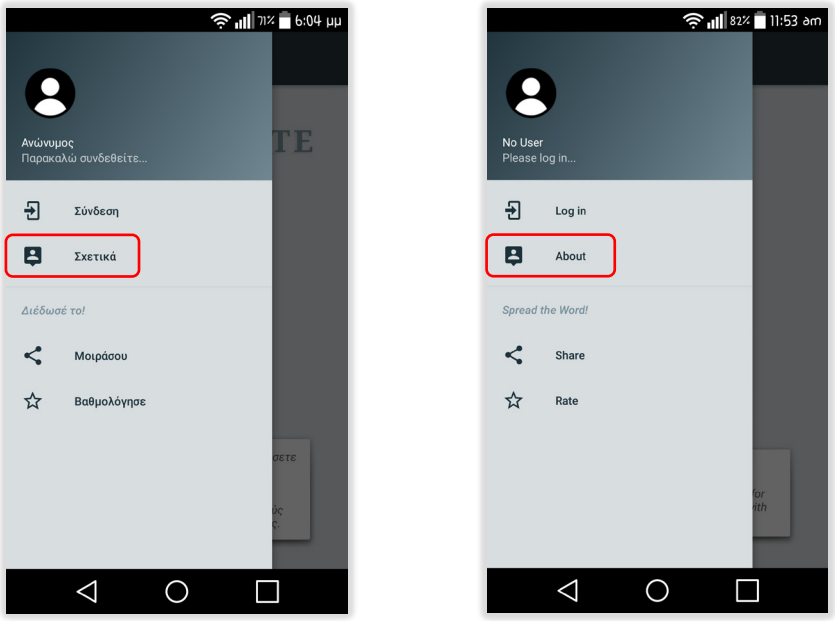
Εάν ο χρήστης αποκριθεί θετικά στο πλαίσιο διαλόγου, τότε ολοκληρώνεται η αποσύνδεσή του από την εφαρμογή και μεταβαίνει στην αρχική οθόνη του επισκέπτη:



Εικόνα 87 Επιτυχής αποσύνδεση χρήστη.

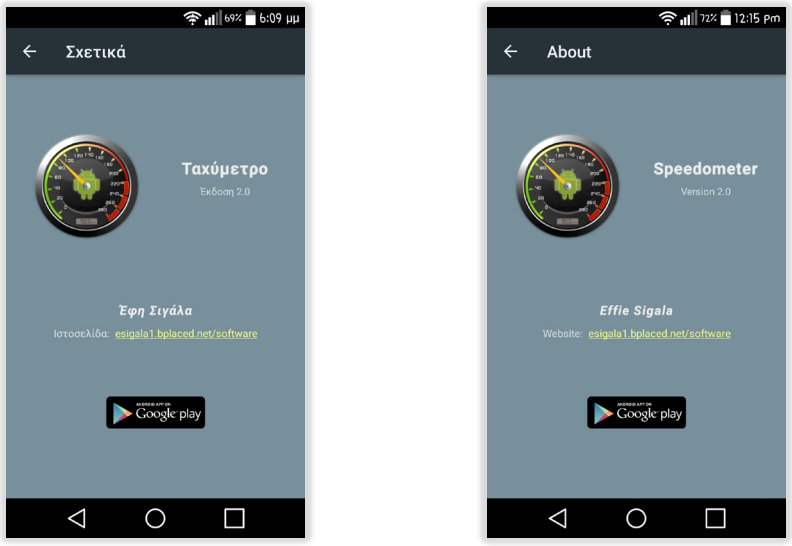
3.2.9 Προβολή Πληροφοριών Εφαρμογής

Όλοι οι τύποι χρηστών (επισκέπτης / πιστοποιημένος χρήστης) μπορούν να δουν τις πληροφορίες της εφαρμογής, επιλέγοντας «Σχετικά» από το συρτάρι πλοήγησης:



Εικόνα 88 Επιλογή «Σχετικά» από το συρτάρι πλοήγησης.

Ο χρήστης μεταβαίνει στην οθόνη «Σχετικά», στην οποία μπορεί να δει πληροφορίες σχετικές με την εφαρμογή (όνομα και έκδοση εφαρμογής, όνομα και ιστοσελίδα προγραμματιστή):

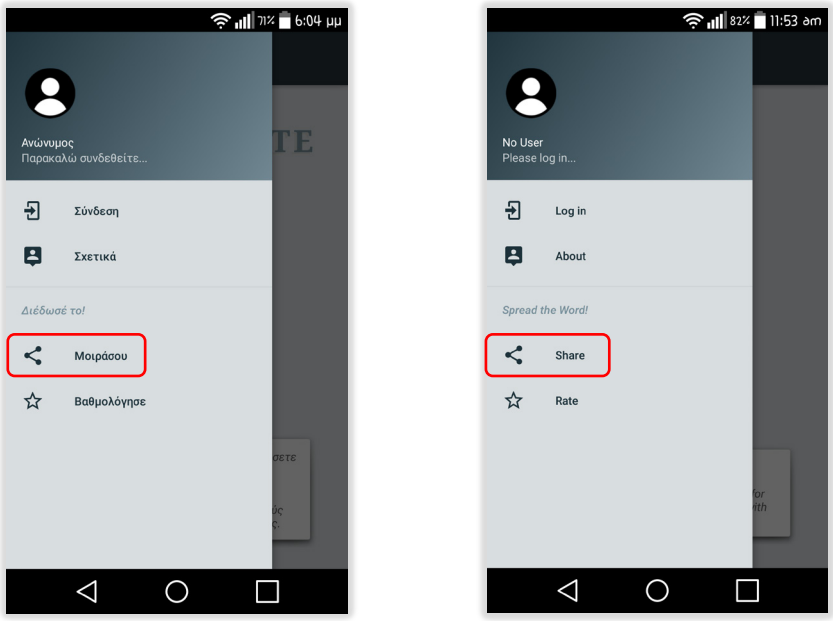


Εικόνα 89 Οθόνη «Σχετικά».

Σημείωση: Πατώντας πάνω στο λογότυπο της εφαρμογής ή πάνω στο λογότυπο του Play Store, ο χρήστης μεταφέρεται στη σελίδα της εφαρμογής στο Play Store.

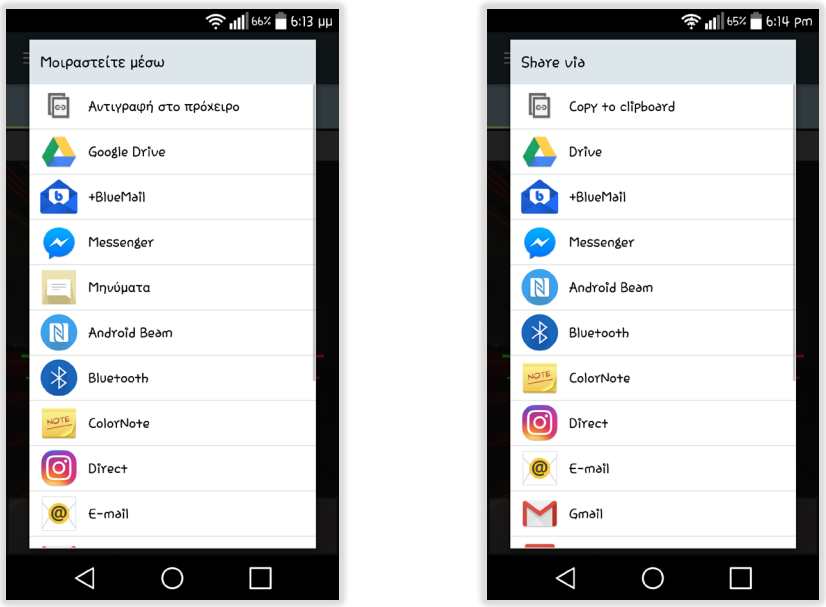
3.2.10 Διαμοιρασμός Εφαρμογής

Όλοι οι τύποι χρηστών (επισκέπτης / πιστοποιημένος χρήστης) μπορούν να κοινοποιήσουν ή να μοιραστούν την εφαρμογή, επιλέγοντας «Μοιράσου» από το συρτάρι πλοήγησης:



Εικόνα 90 Επιλογή «Μοιράσου» από το συρτάρι πλοήγησης.

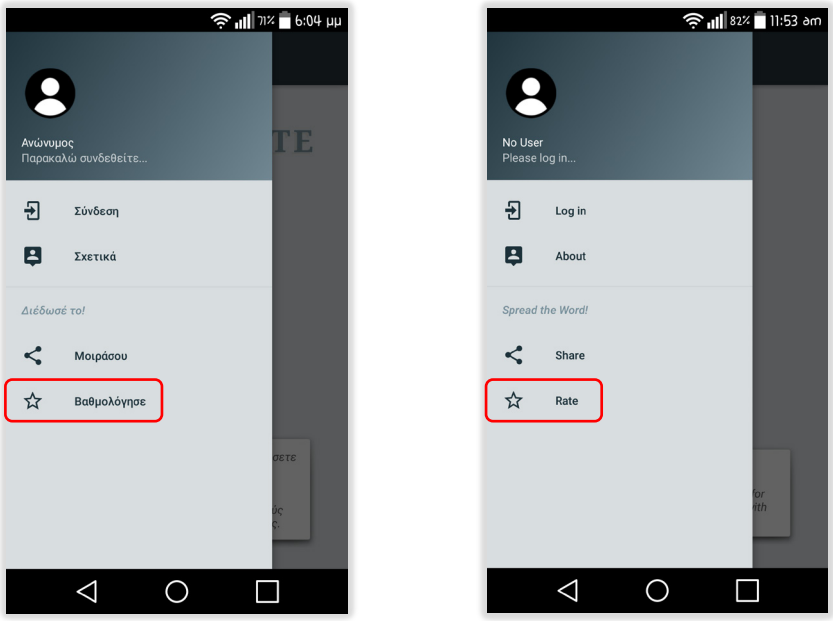
Μέσω ενός παραθύρου διαλόγου, ο χρήστης μπορεί να κοινοποιήσει ή να μοιραστεί την εφαρμογή μέσω ποικίλων μεθόδων. Οι μέθοδοι εξαρτώνται αποκλειστικά από τις εγκατεστημένες εφαρμογές που έχει ο χρήστης στη συσκευή του:



Εικόνα 91 Μέθοδοι διαμοιρασμού της εφαρμογής.

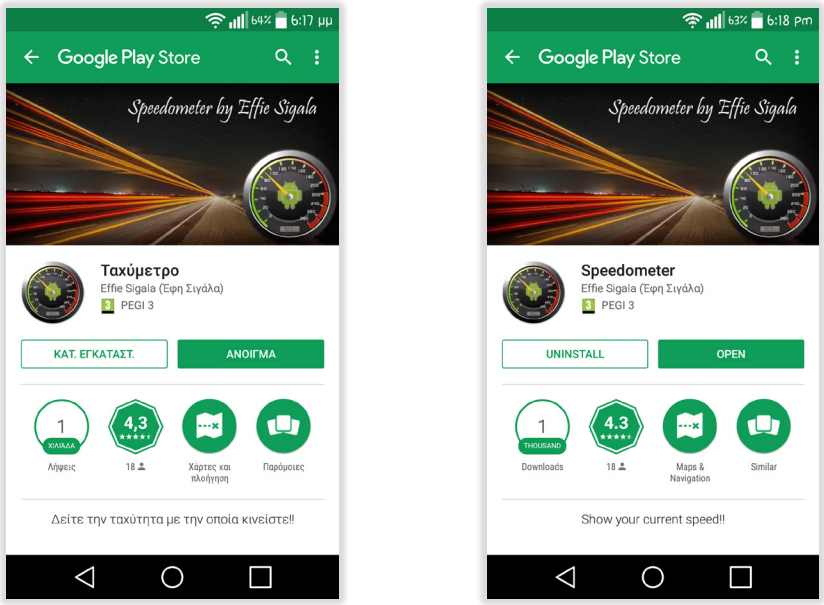
3.2.11 Αξιολόγηση Εφαρμογής στο Play Store

Όλοι οι τύποι χρηστών (επισκέπτης / πιστοποιημένος χρήστης) μπορούν να αξιολογήσουν την εφαρμογή στο Play Store, επιλέγοντας «Βαθμολόγησε» από το συρτάρι πλοήγησης:



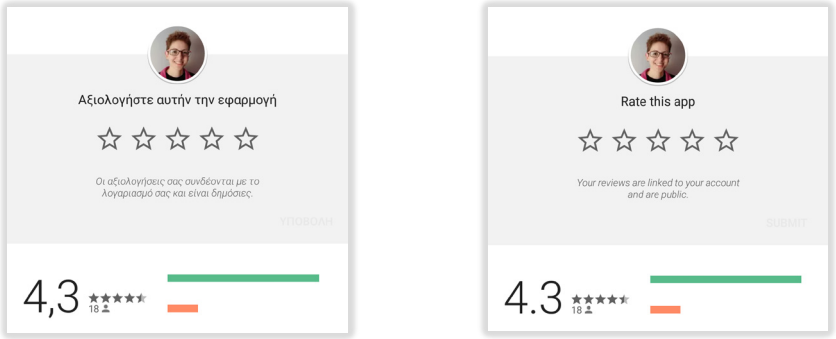
Εικόνα 92 Επιλογή «Βαθμολόγησε» από το συρτάρι πλοήγησης.

Ο χρήστης μεταφέρεται στη σελίδα της εφαρμογής **Ταχύμετρο (Speedometer)** στο Play Store:
https://play.google.com/store/apps/details?id=net.bplaced.esigala1.speedometer



Εικόνα 93 Σελίδα εφαρμογής «Ταχύμετρο» (Speedometer) στο Play Store.

Ολισθαίνοντας προς το μέσον της σελίδας, βρίσκονται οι τρέχουσες αξιολογήσεις της εφαρμογής, όπου ο χρήστης μπορεί να καταχωρήσει τη δικιά του βαθμολογία.



Εικόνα 94 Αξιολόγηση εφαρμογής στο Play Store.

ΚΕΦΑΛΑΙΟ 4: Συμπεράσματα και Μελλοντικές Επεκτάσεις

Οι έξυπνες συσκευές (smartphones) έχουν γίνει αναπόσπαστο κομμάτι της ζωής μας και οι περισσότεροι από εμάς συνηθίζουμε να τις φέρουμε πάνω μας οπουδήποτε κι αν βρισκόμαστε. Οι περισσότερες από αυτές, διαθέτουν ισχυρούς επεξεργαστές και πλήθος αισθητήρων (sensors), όπως ενσωματωμένο σύστημα GPS (Global Positioning System) και αισθητήρα επιταχυνσιόμετρου (accelerometer), δίνοντας στους προγραμματιστές τη δυνατότητα να αναπτύξουν πλήθος εφαρμογών, αξιοποιώντας τα δεδομένα που λαμβάνουν από αυτούς.

Στην παρούσα μεταπτυχιακή διατριβή παρουσιάστηκε μία crowdsensing εφαρμογή, η οποία συλλέγει δεδομένα από τους αισθητήρες των κινητών συσκευών και συγκεκριμένα από έξυπνα κινητά τηλέφωνα και ταμπλέτες, με λειτουργικό σύστημα Android. Η συλλογή των δεδομένων είναι ευκαιριακή, όπου οι χρήστες συμβάλλουν έμμεσα στη συλλογή των πληροφοριών, εκκινώντας απλώς την εφαρμογή.

Η εφαρμογή προσομοιώνει τη λειτουργία ενός ταχύμετρου, χρησιμοποιώντας είτε το ενσωματωμένο σύστημα GPS είτε τα δίκτυα της συσκευής. Υπολογίζει την επιτάχυνση, τη μέση και τη μέγιστη ταχύτητα, τη μέση και τη μέγιστη επιτάχυνση, το συνολικό χρόνο και τη διανυθείσα απόσταση. Επιπρόσθετα, εμφανίζει την τρέχουσα τοποθεσία του χρήστη πάνω στο χάρτη, καθώς και το ιστορικό των διαδρομών του.

Τα δεδομένα που συλλέγονται αφορούν την τοποθεσία του χρήστη, την ταχύτητα, καθώς και τους κραδασμούς που προκαλούνται κατά τη μετακίνησή του, χωρίς όμως την περαιτέρω ανάλυσή τους. Ιδιαίτερο ενδιαφέρον θα αποτελούσε η ανάλυση των δεδομένων αυτών (Big Data), μέσω κατάλληλων τεχνικών (Big Data Analytics), οδηγώντας στην εξαγωγή χρήσιμων πληροφοριών και την παραγωγή χαρτών, σε πραγματικό χρόνο (real-time), με:

- τις συνθήκες κυκλοφοριακής συμφόρησης,
- την ποιότητα του οδοστρώματος ανά περιοχή.

Να σημειωθεί πως ο έλεγχος της ποιότητας και της αξιοπιστίας των δεδομένων είναι αναγκαίος, καθότι δεν αποκλείεται η ύπαρξη πλαστών δεδομένων (counterfeit data) που παραχωρούνται από κακόβουλους χρήστες.

Με την αξιοποίηση της γνώσης που παράγεται από τις εξαγόμενες πληροφορίες, οι χρήστες θα μπορούν να αποφεύγουν τους δρόμους με κυκλοφοριακή συμφόρηση και οι δήμοι να ενημερώνονται αυτόματα για τις κακοτεχνίες του οδοστρώματος, προβαίνοντας σε άμεση αποκατάστασή τους. Οι τεχνικές αυτές έχουν ως απώτερο στόχο τη βελτίωση της μετακίνησης των πολιτών, που συνεπάγεται με τη βελτίωση της ποιότητας της ζωής τους.

Μερικές μελλοντικές επεκτάσεις που αφορούν τη σχεδίαση της εφαρμογής είναι:

- Υποστήριξη περισσοτέρων γλωσσών, εκτός από τα Ελληνικά και τα Αγγλικά που ήδη υποστηρίζει.
- Χρήση τοπικής βάσης δεδομένων, ούτως ώστε τα δεδομένα που συλλέγονται να μην αποστέλλονται απευθείας στη διαδικτυακή βάση δεδομένων, αλλά να επιλέγει ο χρήστης τη χρονική στιγμή και τη μέθοδο του συγχρονισμού των δεδομένων (αυτόματη ή χειροκίνητη).
- Δυνατότητα διαγραφής μιας επιλεγμένης διαδρομής από την τοπική βάση δεδομένων.
- Δυνατότητα κοινοποίησης μιας επιλεγμένης διαδρομής του χρήστη.
- Διατήρηση ιστορικού για τη μέση/μέγιστη ταχύτητα, μέση/μέγιστη επιτάχυνση, συνολικό χρόνο και διανυθείσα απόσταση, ανά ημέρα, δίνοντας στο χρήστη τη δυνατότητα να διαγράψει ή να κοινοποιήσει αυτές τις τιμές.

- Στην προβολή του ιστορικού να δίνεται η δυνατότητα στο χρήστη να επιλέξει τη χρονική στιγμή που τον ενδιαφέρει, προβάλλοντας μία συγκεκριμένη διαδρομή μέσα στην ημέρα.
- Υποστήριξη περισσοτέρων θεμάτων (διαφορετικά χρώματα και γραμματοσειρές), ούτως ώστε η εφαρμογή να εξατομικεύεται για κάθε χρήστη.

Βιβλιογραφικές Αναφορές

Binstock, A. (2015, May 20). *Java's 20 Years of Innovation.* Ανακτήθηκε 13 Οκτωβρίου 2017, από:
https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/

Ducrohet, X. & Norbye, T. & Chou, K. (2013, May 15). *Android Studio: An IDE built for Android.* Ανακτήθηκε 17 Οκτωβρίου 2017, από:
https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html

Echessa, J. (2015, November 4). *Google Play Services for Location and Activity Recognition*. Ανακτήθηκε 19 Οκτωβρίου 2017, από:
https://www.sitepoint.com/google-play-services-location-activity-recognition/

Ganti, R. & Ye, F. & Lei, H. (2011, November 11). *Mobile Crowdsensing: Current State and Future Challenges, IEEE Communications Magazine, Volume 49, Issue 11.* IEEE. DOI: 10.1109/MCOM.2011.6069707.

Gornik, D. (2011, January). *IBM Rational Unified Process: Best Practices for Software Development Teams.* U.S.A.: Rational Software. TP026B.

Jian, A. & Xiaolin, G. & Jianwei, Y. & Yu, S. & Xin, H. (2015, April 20). *Mobile Crowd Sensing for Internet of Things: A Credible Crowdsourcing Model in Mobile-Sense Service.* IEEE. DOI: 10.1109/BigMM.2015.62.

Kruchten, P. (December 2003). *The Rational Unified Process: An Introduction, 3rd Edition.* U.S.: Addison-Wesley Professional. ISBN-13: 978-0-321-19770-2

Sarajlija, S. (2015, August 10). *An Android library for drawing gauges on Canvas.* Ανακτήθηκε 28 Οκτωβρίου 2017, από:
https://github.com/Sulejman/GaugeView

Stacy, A. (2017, June 14). *Reduce friction with the new Location APIs.* Ανακτήθηκε 23 Οκτωβρίου 2017, από:
https://android-developers.googleblog.com/2017/06/reduce-friction-with-new-location-apis.html

Tanas, C. & Herrera, J. (2013, October 24). *When users become sensors: can we trust their readings?* International Journal of Communication Systems, pp: 601-614. ISSN: 10745351. DOI: 10.1002/dac.2689.

Activities. (2016, December 20). Ανακτήθηκε 24 Οκτωβρίου 2017, από:
https://developer.android.com/guide/components/activities/

AsyncTask. (2017, October 25). Ανακτήθηκε 31 Οκτωβρίου 2017, από:
https://developer.android.com/reference/android/os/AsyncTask.html

Build a Responsive UI with ConstraintLayout. (2017, June 14). Ανακτήθηκε 22 Οκτωβρίου 2017, από:
https://developer.android.com/training/constraint-layout

Class LocalBroadcastManager. (2017, August 8). Ανακτήθηκε 24 Οκτωβρίου 2017, από:
https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html

Firebase Pricing Plans. (2017). Ανακτήθηκε 17 Οκτωβρίου 2017, από:
https://firebase.google.com/pricing/

Fragments. (2017, September 26). Ανακτήθηκε 24 Οκτωβρίου 2017, από:
https://developer.android.com/guide/components/fragments.html

Worldwide Sales of Smartphones (2017, May 23). Ανακτήθηκε 28 Οκτωβρίου 2017, από:
https://www.gartner.com/newsroom/id/3725117

Google Sign-In. Ανακτήθηκε 19 Οκτωβρίου 2017, από:
https://developers.google.com/identity/

Implementing Effective Navigation. (2016, November 1). Ανακτήθηκε 25 Οκτωβρίου 2017, από:
https://developer.android.com/training/implementing-navigation/

Introduction to the Google Maps Android API. (2017, June 7). Ανακτήθηκε 19 Οκτωβρίου 2017, από:
https://developers.google.com/maps/documentation/android-api/intro

Making Your App Location-Aware (2016, November 1). Ανακτήθηκε 23 Οκτωβρίου 2017, από:
https://developer.android.com/training/location/

Mobile Crowd Sensing (2017, April). Ανακτήθηκε 31 Οκτωβρίου 2017, από:
http://senda.uab.es/node/15

Mobile Operating System (2017, April). Ανακτήθηκε 27 Οκτωβρίου 2017, από:
https://en.wikipedia.org/wiki/Mobile_operating_system

Overview of Google Play Services. (2017, May 17). Ανακτήθηκε 19 Οκτωβρίου 2017, από:
https://developers.google.com/android/guides/overview

Package android.location. (2017, July 8). Ανακτήθηκε 23 Οκτωβρίου 2017, από:
https://developer.android.com/reference/android/location/package-summary.html

Picasso. (2013). Ανακτήθηκε 17 Οκτωβρίου 2017, από:
http://square.github.io/picasso/

Platform Versions. (2017, October 2). Ανακτήθηκε 22 Οκτωβρίου 2017, από: https://developer.android.com/about/dashboards/index.html#Platform

Services. (2017, August 22). Ανακτήθηκε 24 Οκτωβρίου 2017, από: https://developer.android.com/guide/components/services.html

Settings. (2017, August 21). Ανακτήθηκε 25 Οκτωβρίου 2017, από: https://developer.android.com/guide/topics/ui/settings.html

Simple, battery-efficient location API for Android. Ανακτήθηκε 23 Οκτωβρίου 2017, από: https://developers.google.com/location-context/fused-location-provider/

What is API Level. (2017, June 11). Ανακτήθηκε 22 Οκτωβρίου 2017, από: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels